

Dynamic Memory Design

1991/1992 Data Book/Handbook

© 1991 Advanced Micro Devices, Inc.

Advanced Micro Devices reserves the right to make changes in its products without notice in order to improve design or performance characteristics.

This publication neither states nor implies any warranty of any kind, including but not limited to implied warrants of merchantability or fitness for a particular application. AMD assumes no responsibility for the use of any circuitry other than the circuitry in an AMD product.

The information in this publication is believed to be accurate in all respects at the time of publication, but is subject to change without notice. AMD assumes no responsibility for any errors or omissions, and disclaims responsibility for any consequences resulting from the use of the information included herein. Additionally, AMD assumes no responsibility for the functioning of undescribed features or parameters.

Trademarks

Am386 is a registered trademark of Advanced Micro Devices, Incorporated.

MultiBus is a registered trademark of Intel Corporation.

NeBus is a trademark of Texas Instruments Incorporated.

Macintosh II is a registered trademark of Apple Computer Incorporated.

Micro Channel, PC-AT and PS/2 are trademarks of IBM Corporation.

A D V A N C E D M I C R O D E V I C E S



Dynamic Memory Design

1991/1992 Data Book/Handbook

© 1991 Advanced Micro Devices, Inc.

Advanced Micro Devices reserves the right to make changes in its products without notice in order to improve design or performance characteristics.

This publication neither states nor implies any warranty of any kind, including but not limited to implied warrants of merchantability or fitness for a particular application. AMD assumes no responsibility for the use of any circuitry other than the circuitry in an AMD product.

The information in this publication is believed to be accurate in all respects at the time of publication, but is subject to change without notice. AMD assumes no responsibility for any errors or omissions, and disclaims responsibility for any consequences resulting from the use of the information included herein. Additionally, AMD assumes no responsibility for the functioning of undescribed features or parameters.

Trademarks

Am386 is a registered trademark of Advanced Micro Devices, Incorporated.

Multibus is a registered trademark of Intel Corporation.

NuBus is a trademark of Texas Instruments Incorporated.

Macintosh II is a registered trademark of Apple Computer Incorporated.

Micro Channel, PC-AT and PS/2 are trademarks of IBM Corporation.



ADVANCED MICRO DEVICES

A pioneer in the field of Dynamic Memory Management since 1981, AMD offers a complete solution for the design of today's sophisticated, high-speed memory systems—a family of CMOS DRAM Management building blocks comprising the following:

Dynamic Memory subsystems are increasingly becoming the main system performance bottleneck in low, medium, and high-end computation and communication systems. Performance advances in microprocessor and bus technologies are outpacing memory throughput improvements. AMD's Dynamic Memory Management Products are designed to reduce memory system overhead penalties through high integration. Sub-micron CMOS technology and configurable functionality provide system-specific performance enhancement with low static and dynamic power consumption.

Fred J. Roeder

Fred J. Roeder
Vice President
Standard Products Division

A Product Selector Guide appears on page viii following the Table of Contents.

Chapter 1 gives an overview of the three principle memory management building blocks.

Chapter 2 contains a collection of material to aid the user in designing his memory subsystem. It includes:

- A discussion of DRAM types, special access modes and refresh types.
- Two chapters from Clearpoint Research Corporation's "Designer's Guide to Add-on Memory."
- Error detection and correction system architectures and capabilities.
- And finally, a brief overview of system buses.

Chapter 3 presents five application notes describing different interface designs using the Am29C88 4M Configurable Dynamic Memory Controller/Driver (CDMC).

Chapter 4 comprises two application notes demonstrating the capabilities of the Am29C88 32-bit Error Detection and Correction (EDC) Circuit and the Am29C88 CDMC when used with the IBM PC-AT and PS/2 bus architectures.

Chapter 5 is a collection of article reprints: two detailing the Am29C88 CDMC circuit, another describing a demonstration board using the Am29C71 Video Data Compression/Expansion Processor (VCEP) and the Am29C88 CDMC, and one presenting the ISDN board using the Am29C88 CDMC circuit.

Chapter 6 contains memory management data sheets as listed in the table of contents.

Chapter 7 shows packaging and physical dimensions.

The Appendices are brief discussions of the behavioral simulation models from Logic Automation, Inc. and the electronic design automation tools from Q-CAD.

PREFACE



A pioneer in the field of Dynamic Memory Management since 1981, AMD offers a complete solution for the design of today's sophisticated, high-speed memory systems—a family of CMOS DRAM Management building blocks comprising the following:

- Dynamic Memory Controllers
- Error Detection and Correction (EDC) Circuits
- Multiple Bus Exchange (MBE)
- DRAM Drivers

These versatile memory management circuits can ease the design task precipitated by new and faster microprocessors, including RISC microprocessors, the increased demands for more system memory, and the requirements arising when designing DRAMs into smaller systems. They also offer design flexibility for expanding the basic 16-bit system to 32- or 64-bit word widths, and beyond.

This handbook/data book provides descriptions of the memory management circuits including specifications and gives specific examples of how to design dynamic memory systems, using these high-speed CMOS building blocks. All necessary functions are available to the system designer so that he can obtain the best cost/performance ratio to satisfy his memory-system design.

A Product Selector Guide appears on page viii following the Table of Contents.

Chapter 1 gives an overview of the three principle memory management building blocks.

Chapter 2 contains a collection of material to aid the user in designing his memory subsystem. It includes:

- A discussion of DRAM types, special access modes and refresh types,
- Two chapters from Clearpoint Research Corporation's "Designer's Guide to Add-on Memory,"
- Error detection and correction system architectures and capabilities,
- And finally, a brief overview of system buses.

Chapter 3 presents five application notes describing different interface designs using the Am29C688 4M Configurable Dynamic Memory Controller/Driver (CDMC).

Chapter 4 comprises two application notes demonstrating the capabilities of the Am29C660 32-bit Error Detection and Correction (EDC) Circuit and the Am29C668 CDMC when used with the IBM PC-AT and PS/2 bus architectures.

Chapter 5 is a collection of article reprints: two detailing the Am29C668 CDMC circuit, another describing a demonstration board using the Am95C71 Video Data Compression/Expansion Processor (VCEP) and the Am29C668 CDMC, and one presenting the ISDN board using the Am29C668 CDMC circuit.

Chapter 6 contains memory management data sheets as listed in the table of contents.

Chapter 7 shows packaging and physical dimensions.

The Appendices are brief discussions of the behavioral simulation models from Logic Automation, Inc. and the electronic design automation tools from OrCAD.



TABLE OF CONTENTS

Chapter 1—Dynamic Memory Design Overview	1-1
AMD's System Design Methodology	1-3
Family Overview	1-4
Dynamic Memory Control	1-4
Error Detection and Correction (EDC)	1-4
Multiple Bus Exchange (MBE)	1-4
DRAM Driver	1-4
Chapter 2—Memory System Architectures	2-1
Introduction	2-3
DRAM Basics	2-4
DRAM Types and Accesses	2-4
DRAM Refresh Types	2-7
Understanding Memory Design	2-9
For the Memory Novice: An Overview	2-9
Memory Design for Improved System Performance	2-12
Bus Efficiency	2-12
Memory Technologies and High Performance	2-14
New Technology	2-17
Design Integrity	2-18
Designing for Reliability	2-20
The Probability of Errors	2-20
Error Protection	2-20
Running Bare	2-21
Parity Generation and Checking	2-21
Error Detection and Correction	2-21
The Mechanics of EDC	2-21
Parity vs EDC: A Comparison	2-22
The Importance of the EDC Word Length	2-22
EDC on Array Cards	2-22
Redundancy	2-23
Mean Time to Repair	2-24
References and Site Visits	2-24
The Bottom Line	2-24
Error Detection and Correction Architectures	2-25
Fly-By	2-25
Flow-Through	2-26
Refresh With Scrubbing	2-26
Am29C660 CMOS Cascadable 32-Bit EDC Circuit	2-28
Program to Evaluate Am29C660 Multiple Error Detection Capability	2-28
Memory Reliabilities with and without the 29C660 EDC Circuit	2-31
System Buses	2-33
VMEbus	2-33
Multibus I and II	2-33
Nu Bus	2-33
AT Bus	2-34
Micro Channel	2-34

EISA	2-34
Q-Bus	2-34
Chapter 2—Memory System Architectures (Continued)	
MicroVAX II	2-34
MicroVAX 3000	2-35
Futurebus+	2-35
Chapter 3—Microprocessor Interfaces to the Am29C668 Configurable Dynamic	
Memory Controller/Driver	3-1
Introduction	3-3
Am29C668 CDMC to Am29000 Streamlined Instruction	
Processor Interface	3-5
Am29C668 CDMC to 80C286 Microprocessor Interface	3-41
Am29C668 CDMC to Am386™ Microprocessor Interface	3-57
Am29C668 CDMC to 68020 Microprocessor Interface	3-73
Am29C668 CDMC to 80486 Microprocessor	3-93
Chapter 4—EDC Memory Board Designs	
Introduction	4-1
IBM PC-AT Plug-in Memory Card with EDC	4-2
IBM PC-AT Plug-in Memory Card with EDC	4-3
IBM PS/2 12-Mbyte Memory Board with EDC	4-25
Chapter 5—Special Applications and Article Reprints	
Introduction	5-1
Configurable DRAM Controller Enhances System Performance	5-2
Four Megabit DRAM Controller Offers Burst Addressing	5-3
High-Speed VCEP Demonstration Board Using the	
Am29C668 CDMC	5-11
Universal Coprocessor Platform (UCP) and Network Terminator-S	
Interface (NTs)	5-15
Interface (NTs)	5-19
Chapter 6—Product Data Sheets	
Am2965/Am2966 Octal Dynamic Memory Drivers with Three-State	
Outputs	6-1
Am29C60A CMOS Cascadable 16-Bit Error Detection	
and Correction Unit	6-3
Am29C660/A/B/C/D/E CMOS Cascadable 32-Bit Error Detection and	
Correction Circuit	6-12
Am29C668 4M Configurable Dynamic Memory Controller/Driver	6-29
Am29C676 11-Bit DRAM Driver	6-69
Am29C827A/Am29C828A High-Performance CMOS	
Bus Buffers	6-112
Am29C983/Am29C983A 9-Bit x 4-Port Multiple Bus Exchange	6-133
Am29C985 9-Bit x 4-Port Multiple Bus Exchange with Parity	6-143
Am29C985 9-Bit x 4-Port Multiple Bus Exchange with Parity	6-156
Chapter 7—Physical Dimensions	
Appendix A—Behavioral Simulation Models from Logic Automation, Inc.	7-1
Appendix B—Electronic Design Automation Tools From	
OrCAD Systems Corporation	A-3
Appendix B—Electronic Design Automation Tools From	
OrCAD Systems Corporation	B-3

PRODUCT SELECTOR GUIDE



DRAM Controllers

Am29C668	4M Configurable Dynamic Memory Controller/Driver, 34 ns	6-69
Am29C668-1	4M Configurable Dynamic Memory Controller/Driver, 29 ns	6-69

Error Detection and Correction Circuits

Am29C660	32-Bit CMOS Cascadable EDC (40 ns Error Detect)	6-29
Am29C660A	32-Bit CMOS Cascadable EDC (30 ns Error Detect)	6-29
Am29C660B	32-Bit CMOS Cascadable EDC (25 ns Error Detect)	6-29
Am29C660C	32-Bit CMOS Cascadable EDC (16 ns Error Detect)	6-29
Am29C660D	32-Bit CMOS Cascadable EDC (12 ns Error Detect)	6-29
Am29C660E	32-Bit CMOS Cascadable EDC (9 ns Error Detect)	6-29
Am29C60A	16-Bit CMOS Cascadable EDC (20 ns Error Detect)	6-20

Multiple Bus Exchanges

Am29C983	9-Bit x 4-Port MBE	6-143
Am29C983A	9-Bit x 4-Port MBE, High Speed	6-143
Am29C985	9-Bit x 4-Port MBE with Parity	6-155

DRAM Drivers

Am2966	8-Bit DRAM Driver, Non-inverting	6-3
Am2965	8-Bit DRAM Driver, Inverting	6-3
Am29C676	11-Bit DRAM Driver	6-112
Am29C827A	10-Bit CMOS Non-inverting Buffer	6-133
Am29C828A	10-Bit CMOS Inverting Buffer	6-133



CHAPTER 1

Dynamic Memory Design Overview

AMD's System Design Methodology	1-3
Family Overview	1-4
Dynamic Memory Control	1-4
Error Detection and Correction (EDC)	1-4
Multiple Bus Exchange (MBE)	1-4
DRAM Drivers and Buffers	1-4



CHAPTER 1 Dynamic Memory Design Overview

1-3	AMD's System Design Methodology
1-1	Family Overview
1-4	Dynamic Memory Control
1-1	Error Detection and Correction (EDC)
1-1	Multiple Bus Exchange (MBE)
1-1	DRAM Drivers and Buffers



Dynamic Memory Design Overview

AMD's SYSTEM DESIGN METHODOLOGY

AMD's new CMOS Dynamic Memory Management family, featuring maximum performance and flexibility, offers a complete system solution for memory-system design. This family contains functions that are generally applicable for a wide range of memory requirements over the entire computing spectrum, from powerful desktop PC's and workstations through superminis, mainframes and telecommunication applications. In each system area, the AMD solution achieves maximum performance and reliability at minimum cost and with minimum device count.

All necessary functions are available to the system designer so that he can tailor the memory subsystem to his specific requirements:

- Complete address-path and refresh control,
- Controlled edge-rate drive for DRAM address and strobe inputs,
- Automatic DRAM access timing,
- Error Detection and Correction (EDC),
- System data-bus interface.

The AMD DRAM Management building blocks offer design flexibility in a variety of applications—expansion from the basic 16-bit system to 32- or 64-bit systems, and beyond, with or without EDC protection.

Today's high-performance "burst" microprocessors running at speeds above 16 MHz are fully supported. Read/write features include full support of byte writing, selectable access-timing options, burst-mode access support, page-mode-access support, static-column-access support, true bank interleaving, and selectable output-drive configurations. Using AMD's proprietary cache-access mode, designers can take full advantage of the performance benefits of page-mode DRAMs by continually comparing bank and row addresses during subsequent accesses.

Selectable refresh options include standard row refresh and CAS-before-RAS refresh, which is supported by some DRAMs. In a system employing EDC logic for memory-system integrity, refresh with scrubbing that prevents accumulation of soft errors is fully supported. Configurable row, column, and bank refresh counters and timing logic provide for a built-in EDC initialization mode, during which a known value is written to every memory location before starting normal operation.

Newer members of the Dynamic Memory Management family are processed using AMD's advanced submicron CMOS technology. Full qualification and reliability data is available upon request.

The driving force behind this application handbook/data book is the determination to provide immediate answers to most common memory design and application questions. Full application support is vital throughout the complete design cycle from conception, through working prototypes, to production.

FAMILY OVERVIEW

Dynamic Memory Control

AMD's newest and most sophisticated DRAM controller is the Am29C668 4M Configurable Dynamic Memory Controller/Driver. This device provides the logic necessary to access and refresh 64K, 256K, 1M and 4M x n DRAMs. New features of the Am29C668 include support for burst-mode microprocessor accessing, automatic access timing, support for page-mode DRAMs, and selectable output-drive configurations. The Am29C668 can directly drive two banks of 39 DRAMs (32-bit word plus seven check bits) or four banks of 22 DRAMs (16-bit word plus six check bits) with its proprietary low-ground-bounce, low-undershoot outputs.

Error Detection and Correction (EDC)

AMD's newest EDC circuit is the CMOS Cascadable 32-bit Am29C660 that uses an industry-standard modified Hamming code to generate check bits and detect and correct hard and soft errors. The Am29C660 may be used with any memory technology including DRAM, SRAM, EPROM, Flash, and other types that exhibit increased soft error rates due to very small cell geometries.

The Am29C660 is currently the world's fastest and lowest power 32-bit EDC circuit. It is currently available in six speed grades; the fastest detects errors in 9 ns and corrects them in 14 ns maximum under worst-case operation conditions. The device is available in industry-standard 68-pin PLCC, 80-pin PQFP and ceramic PGA packaging in both commercial and military versions.

A full range of CMOS 16-bit EDC circuits is also available. These are in the industry standard Am29C60 EDC family. The Am29C60A detects errors in 20 ns and corrects them in 25 ns maximum. It requires the lowest power in the industry.

Multiple Bus Exchange (MBE)

These devices are general-purpose, high-speed, digital cross-point switches, designed to improve interbus communications. The Am29C983 and Am29C983A are 9-bit x 4-port MBEs with input and output latches on all TTL compatible I/O ports. Any port may serve as either a source or destination. Differing sets of two I/Os may communicate concurrently with one another. All outputs have 48 mA drive capability for efficiently driving high capacitive and inductive buses. The Am29C985 9-bit x 4-port MBE incorporates parity-check and generation capabilities. More detailed application information on the MBE may be obtained from the Multiple Bus Exchange Handbook/Data book (PID #10351B).

DRAM Drivers

All of these devices offer proprietary edge-rate-controlled outputs to reduce output undershoots, overshoots and ground bounce. Skew times between outputs are also minimized. The new 11-bit Am29C676 is ideal for driving the eleven address lines of 4 Mbit x 1 and 4 Mbit x 4 DRAMs. The Am29C827A and Am29C828A are 48 mA general purpose bus buffers that may also be used to drive DRAM address and control inputs. These 10-bit wide devices are well suited for driving 1-Mbit x 1 and 1-Mbit x 4 DRAMs. The Am2965 and Am2966 are 8-bit DRAM drivers with industry-standard pinouts.



CHAPTER 2

Memory System Architectures

Introduction	2-3
DRAM Basics	2-4
DRAM Types and Accesses	2-4
DRAM Refresh Types	2-7
Understanding Memory Design	2-9
For the Memory Novice: An Overview	2-9
Memory Design for Improved System Performance	2-12
Bus Efficiency	2-12
Memory Technologies and High Performance	2-14
New Technology	2-17
Design Integrity	2-18
Designing for Reliability	2-20
The Probability of Errors	2-20
Error Protection	2-20
Running Bare	2-21
Parity Generation and Checking	2-21
Error Detection and Correction	2-21
The Mechanics of EDC	2-21
Parity vs. EDC: A Comparison	2-22
The Importance of the EDC Word Length	2-22
EDC on Array Cards	2-22
Redundancy	2-23
Mean Time to Repair	2-24
References and Site Visits	2-24
The Bottom Line	2-24
Error Detection and Correction System Architectures	2-25
Fly-By	2-25
Flow-Through	2-26
Refresh With Scrubbing	2-26
Am29C660 CMOS Cascadable 32-Bit EDC Circuit	2-28
Program to Evaluate Am29C660 Multiple Error Detection Capability	2-28
Memory Reliabilities with and without the Am29C660 EDC Circuit	2-31
System Buses	2-33
VMEbus	2-33
Multibus I and II	2-33
Nu Bus	2-33
AT Bus	2-34
Micro Channel	2-34
EISA	2-34
Q-Bus	2-34
MicroVAX II	2-34
MicroVAX 3000	2-35
Futurebus+	2-35

CHAPTER 2

Memory System Architectures

2-3	Introduction
2-4	DRAM Basics
2-4	DRAM Types and Accesses
2-7	DRAM Refresh Types
2-9	Understanding Memory Design
2-9	For the Memory Novice: An Overview
2-12	Memory Design for Improved System Performance
2-12	Bus Efficiency
2-14	Memory Technologies and High Performance
2-17	New Technology
2-18	Design Integrity
2-20	Designing for Reliability
2-20	The Probability of Errors
2-20	Error Protection
2-21	Running Bits
2-21	Parity Generation and Checking
2-21	Error Detection and Correction
2-21	The Mechanics of EDC
2-22	Parity vs. EDC: A Comparison
2-22	The Importance of the EDC Word Length
2-22	EDC on Array Cards
2-23	Redundancy
2-24	Mean Time to Repair
2-24	References and Site Visits
2-24	The Bottom Line
2-25	Error Detection and Correction System Architectures
2-25	Fly-By
2-26	Flow-Through
2-26	Refresh With Scrubbing
2-28	Am28C680 CMOS Cascadable 32-Bit EDC Circuit
2-28	Program to Evaluate Am28C680 Multiple Error Detection Capability
2-31	Memory Reliability with and without the Am28C680 EDC Circuit
2-33	System Buses
2-33	VMEbus
2-33	Multibus I and II
2-33	Nu Bus
2-34	AT Bus
2-34	Micro Channel
2-34	EISA
2-34	Q-Bus
2-34	MicroVAX II
2-35	MicroVAX 3000
2-35	Futurebus+

Memory System Architectures



INTRODUCTION

Dynamic memory systems differ extensively; they use different types and densities of DRAMs with varying access modes, timing requirements, refresh options, and architectural organizations. The DRAMs are organized in different word lengths, and may support parity or error detection and correction (EDC) with additional memory overhead. Different board layouts, control circuitry, packaging, and bus protocols are also used.

The memory-subsystem design is directly related to the price/performance of the entire computer system. Low-end machines generally have the main memory located on the motherboard. They provide for add-on memory that is accessed by a local memory bus or the system backplane. High-end systems often have separate memory boards that may be added in large quantities depending upon the required configuration.

This chapter contains a collection of material intended to give the new memory designer, as well as the seasoned professional, information to help make memory-subsystem design easier. There is a section on DRAM basics including discussions of special DRAM types, special access modes, and refresh types.

Two sections are chapters selected from *The Designer's Guide to Add-on Memory* from Clearpoint Inc. "Understanding Memory Design" covers the fundamentals of how data is accessed and stored, the different system and component technologies available to accomplish this now and in the future, and a general overview of system integrity. "Designing for Reliability" explains the different options available for detecting and correcting hard and soft memory errors.

Three related industry trends strongly support the case for increased EDC capability in today's high-performance systems, from desktop machines through mainframes. The first trend arises from the need to support sophisticated operating systems and applications software. The result is a requirement for more dynamic memory. With each additional bank of memory, there is an increased probability of soft errors, which increases the need for system protection.

The second trend is driven more by the business aspects of computer design. To remain competitive in today's marketplace, data-processing system require higher density DRAMs to provide more data storage capacity in the same or less amount of real estate, at reasonable prices. As memory-feature sizes are decreased to meet higher density requirements, the probability of both hard and soft errors increases; smaller capacitive cells are more susceptible to bit complementing due to alpha-particle bombardment and electrical noise.

The third and last trend emanates from the user's desire to have a reliable system: one that crashes only rarely, if at all. Today's computer-literate consumers are demanding the security provided by EDC to prevent the microprocessor from attempting to execute or transfer erroneous instructions or data. This is particularly critical in today's "client-server" networked configurations where a memory crash may interrupt dozens of active users.

DRAM BASICS

DRAMs store data in 1-bit cells. One or more cells may be accessed in one data transfer, depending on the organization of the DRAM. Cells are arranged in square grids, with each cell having a specific row/column grid position identified by a bit address consisting of a row address and a column address.

DRAM row/column addresses are multiplexed on one set of address pins. Row addresses are latched on the falling edge of a Row Address Strobe $\overline{\text{RAS}}$ and column addresses are latched on the falling edge of a Column Address Strobe $\overline{\text{CAS}}$. A Write Enable $\overline{\text{WE}}$ signal is used to indicate whether a cycle is Read or Write: Low during Write cycles, High during Read cycles.

DRAM cells are capacitors while static RAMs store bits in transistor cells, where voltages change only during Write cycles. Thus, voltages are *static*, hence the term *static RAM*. On the other hand, voltage levels "leak away" over time from DRAM capacitor cells, which therefore require refreshing at regular intervals to maintain adequate voltage levels. The term *dynamic RAM* comes from this constantly changing cell voltage.

Although static RAM access times are faster and interfaces are easier to design, DRAMs offer the clear advantages of small cell size (and thus higher density), small package size, and lower cost.

DRAM Types and Accesses

There are a number of special-access DRAMs available that help reduce memory access time when used in a particular access mode. The special-access mode is a feature available to the user in addition to the normal RAS/MSEL/CAS fundamental accesses. The Multiplexer Select MSEL is a dual-function input to a DRAM controller, used to determine whether the address to the DRAM is a row or column address.

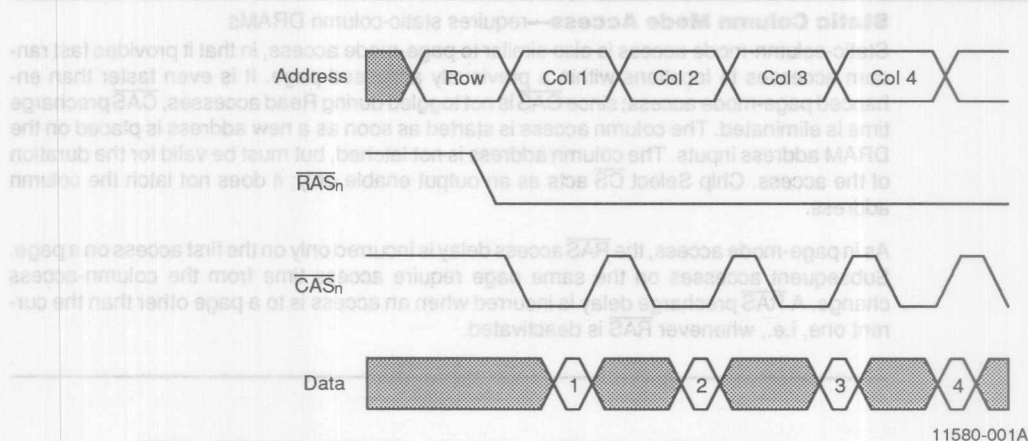
Special-access DRAMs normally command a premium price. However, they can more than compensate for this by appreciably reducing the memory cycle time and enhancing the system performance. The basic choice is dictated by the system configuration and its application, the main objective being enhancement of the overall system performance at a given cost.

The following special-access DRAMs are discussed here:

- Page Mode
- Enhanced Page Mode or Fast Page Mode
- Static Column Mode
- Nibble or Ripple Mode

Page-Mode Access—performed with regular DRAMs

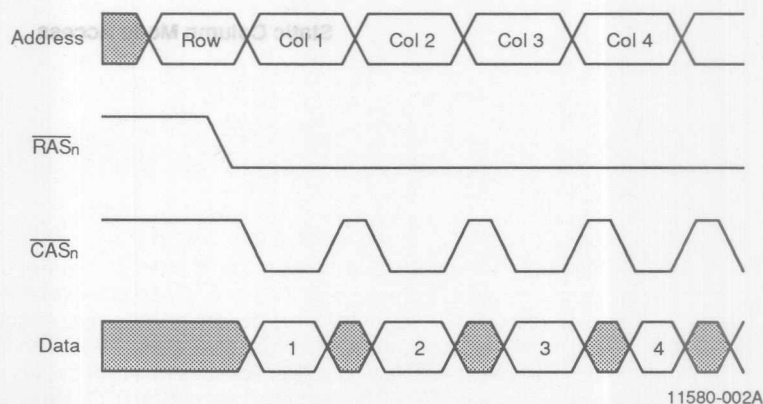
Page-mode access provides for fast random access of locations within a page, i.e., DRAM row, by saving the $\overline{\text{RAS}}$ precharge time for every access within the page. It starts as a normal access with a $\overline{\text{RAS}}$ time for the initial access on the page. All subsequent accesses on the same page require only the assertion of the $\overline{\text{CAS}}$ input. At the end of the initial access, $\overline{\text{CAS}}$ is deactivated while $\overline{\text{RAS}}$ is held active. For subsequent accesses within the page, a new column address is placed on the address inputs of the DRAM and $\overline{\text{CAS}}$ is asserted, thus initiating the page-mode access. The access time is calculated from the active edge of $\overline{\text{CAS}}$. A $\overline{\text{RAS}}$ precharge delay is only incurred for accesses on pages other than the current one, i.e., whenever $\overline{\text{RAS}}$ is deactivated. Page-mode accesses may be non-sequential; i.e., as long as the row address is unchanged, any column address may be selected in any order.



Page-Mode Access

Enhanced Page-Mode Access—requires page mode DRAMs

Enhanced page-mode access is similar to page-mode access, in that it provides fast random accesses to locations within a page by eliminating the \overline{RAS} precharge time for page accesses after the initial access. However, it is faster than a standard page-mode access because the next access is started as soon as a new column is placed on the DRAM address lines, rather than starting from the \overline{CAS} active time. \overline{CAS} still latches the column data and acts as an output enable, but the page access starts from the column address change, rather from the active edge of \overline{CAS} .

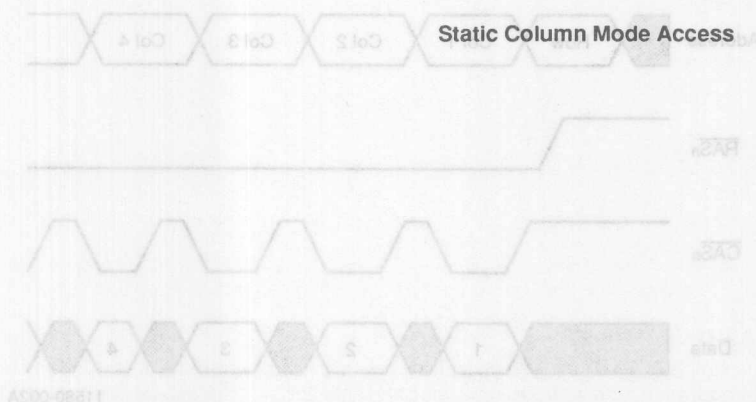
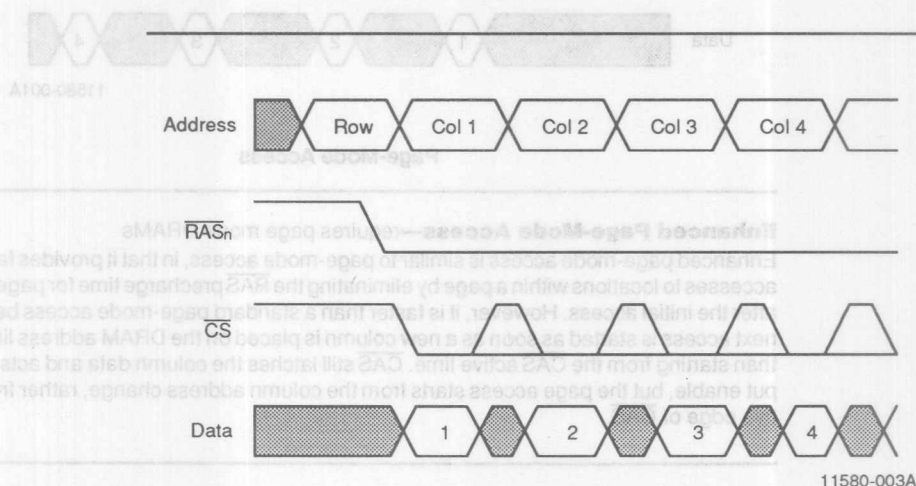


Enhanced Page-Mode Access

Static Column Mode Access—requires static-column DRAMs

Static-column-mode access is also similar to page-mode access, in that it provides fast random accesses to locations within a previously accessed page. It is even faster than enhanced page-mode access; since $\overline{\text{CAS}}$ is not toggled during Read accesses, $\overline{\text{CAS}}$ precharge time is eliminated. The column access is started as soon as a new address is placed on the DRAM address inputs. The column address is not latched, but must be valid for the duration of the access. Chip Select $\overline{\text{CS}}$ acts as an output enable only; it does not latch the column address.

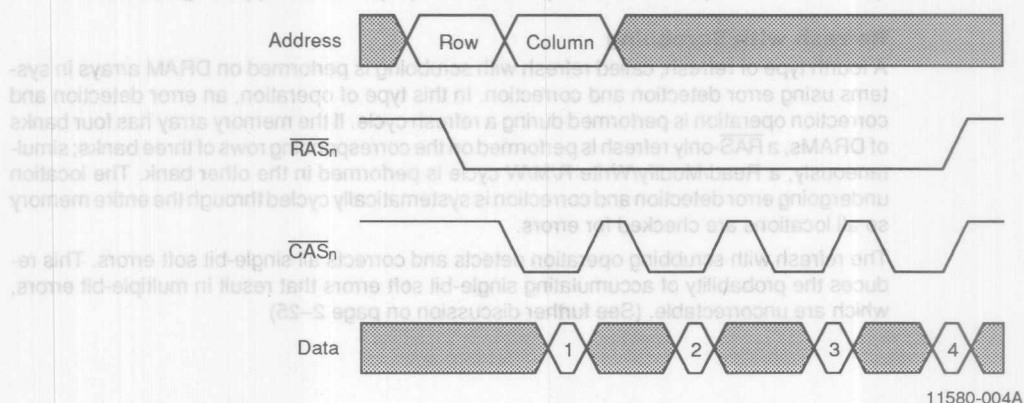
As in page-mode access, the $\overline{\text{RAS}}$ access delay is incurred only on the first access on a page. Subsequent accesses on the same page require access time from the column-access change. A $\overline{\text{RAS}}$ precharge delay is incurred when an access is to a page other than the current one, i.e., whenever $\overline{\text{RAS}}$ is deactivated.



Nibble-Mode Access—requires nibble-mode DRAMs

The nibble-mode access provides for fast random access of four locations in modulo-4 order, i.e., 2, 3, 4, 1 or 4, 1, 2, 3, etc., with only one address from the system. The remaining addresses are generated internally by the DRAM. This frees up the address bus while the memory is being accessed. The falling edge of CAS initiates the next access.

As in the page-mode access, the $\overline{\text{RAS}}$ access delay is incurred only on the first access of the nibble; the subsequent three accesses require only CAS access time. A $\overline{\text{RAS}}$ precharge delay is incurred between nibble accesses, i.e., whenever $\overline{\text{RAS}}$ is deactivated.



Nibble Mode Access

DRAM Refresh Types

To maintain data integrity, i.e., prevent bits from changing state, all DRAMs must be refreshed within a fixed time, usually 4 ms. Hence, all rows need to be accessed at least once in 4 ms. Refreshing a DRAM row refreshes all the locations in that row.

Interleaved and Burst Refreshes

DRAM refreshes may be interleaved between memory accesses every so often to meet the above condition. This is called interleaved refresh. Another option, called burst refresh, is to refresh all the locations in a continuous burst before the maximum time between refreshes.

An intermix of the above operations may also be performed, in which case a fixed number of burst refresh cycles may be performed between fixed intervals of time.

RAS-Only Refresh

The simplest type of refresh operation, called RAS-only refresh, is performed by placing the row address on the address input lines and activating $\overline{\text{RAS}}$. It can be performed on all types of DRAMs. All the banks of the DRAM array can be refreshed simultaneously using this method.

When operating on more than one bank of DRAMs, the $\overline{\text{RAS}}$ inputs of all the banks can be staggered by a clock cycle. This type of refresh timing is called staggered refresh timing. Staggered refresh helps reduce ground bounce and overshoot/undershoot generation associated with driving high-capacitive and inductive DRAM loads. It requires less power than refreshing all banks at once.

CAS-Before-RAS Refresh

Using CAS-before-RAS refresh, the row-refresh address is generated internally by the DRAM rather than generated by an external DRAM controller. The active edge of $\overline{\text{CAS}}$ increments the on-chip refresh counter; $\overline{\text{RAS}}$ then initiates the actual refresh operation. This type of a refresh operation can be performed only by DRAMs supporting this feature.

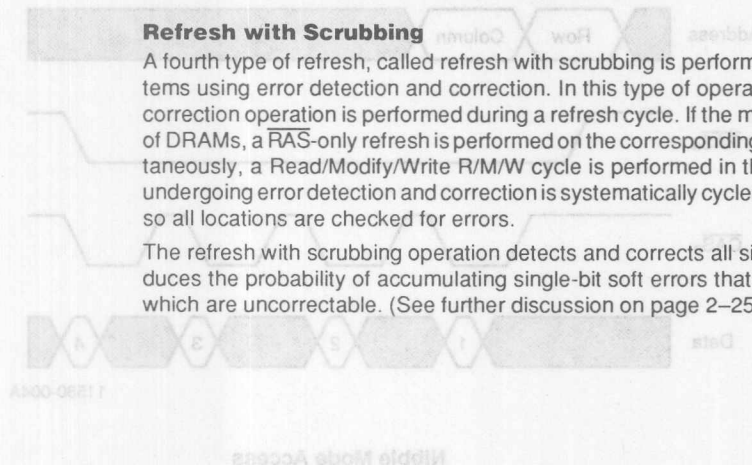
Hidden Refresh

A third type of refresh is called a hidden refresh. The $\overline{\text{CAS}}$ signal holds the data active from a memory access while a row-refresh address is placed on the address inputs and a $\overline{\text{RAS}}$ signal is activated to perform a refresh. Hidden refresh has minimum system impact, since all or most of the refresh cycle is overlapped with an access to another memory or I/O device. This type of a refresh operation can be performed only by DRAMs supporting this feature.

Refresh with Scrubbing

A fourth type of refresh, called refresh with scrubbing is performed on DRAM arrays in systems using error detection and correction. In this type of operation, an error detection and correction operation is performed during a refresh cycle. If the memory array has four banks of DRAMs, a $\overline{\text{RAS}}$ -only refresh is performed on the corresponding rows of three banks; simultaneously, a Read/Modify/Write R/M/W cycle is performed in the other bank. The location undergoing error detection and correction is systematically cycled through the entire memory so all locations are checked for errors.

The refresh with scrubbing operation detects and corrects all single-bit soft errors. This reduces the probability of accumulating single-bit soft errors that result in multiple-bit errors, which are uncorrectable. (See further discussion on page 2-25)



DRAM Refresh Types

To maintain data integrity, i.e., prevent data from changing state, all DRAMs must be refreshed within a fixed time, usually 4 ms. Hence, all rows need to be accessed at least once in 4 ms. Refreshing a DRAM row refreshes all the locations in that row.

Interleaved and Burst Refreshes

DRAM refreshes may be interleaved between memory accesses every so often to meet the above condition. This is called interleaved refresh. Another option, called burst refresh, is to refresh all the locations in a continuous burst before the maximum time between refreshes. An interleaved burst refresh may also be performed, in which case a fixed number of burst refresh cycles may be performed between fixed intervals of time.

RAS-Only Refresh

The simplest type of refresh operation, called RAS-only refresh, is performed by placing the row address on the address input lines and activating $\overline{\text{RAS}}$. It can be performed on all types of DRAMs. All the banks of the DRAM array can be refreshed simultaneously using this method.

When operating on more than one bank of DRAMs, the $\overline{\text{RAS}}$ inputs of all the banks can be staggered by a clock cycle. This type of refresh timing is called staggered refresh timing. Staggered refresh helps reduce ground bounce and overshoot/undershoot generation associated with driving high-capacitive and inductive DRAM loads. It requires less power than refreshing all banks at once.

Understanding Memory Design

The performance of memory is a function of many things, some related to the memory board design and some not. The purpose of this section is to identify the system design elements that affect the performance of memory. While providing a brief overview of how systems operate with memory, this section will also focus on the broad category of bus efficiency. This explanation of the basic elements of memory will include a description of major design issues such as the width of the data bus, the clock speed, bus protocols, array vs. on-board memory control, and ECL (emitter coupled logic) vs. TTL (transistor-transistor logic).

FOR THE MEMORY NOVICE: AN OVERVIEW

In simple terms, the bus on a computer system is a set of electrical connectors or wires attached to the computer backplane (printed circuit board, generally in the back of the computer cabinet). The bus connects to the various elements of the computer system (Central Processing Unit, system memory, disk storage, printers, terminals, etc.) to allow the transfer of electrical signals between the different parts of the system. Typical transactions include the transfer of instructions from the CPU to read data from memory (find it for processing by CPU); to write data (take processed data from CPU back to memory) or to enter data from a terminal or transfer it to disk storage.

The term "bus" is often used broadly, to mean both the hardware (the physical connectors and cable on the computer's backplane) as well as any resulting design constraints (the size of the data bus or the bus protocol). The bus hardware is usually made up of electrical "paths" — designated pins on the connectors — dedicated to transferring data, addresses and/or clock signals along these "paths."

The signal is transmitted according to one of two major timing methods, or protocols: synchronous or asynchronous. The choice of bus protocol determines which timing method is used by the CPU in recognizing electrical signal changes on the bus, thereby coordinating when the various parts of the computer "communicate."

Depending on the bus width and clock cycle time, data is transferred at varying rates. The total configuration of bus width, cycle time and protocol is the major determinant of bus performance.

This chapter explains these elements and how they impact memory performance.

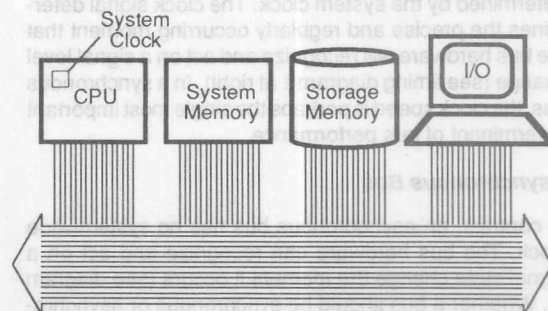
Width of the Bus

The width of the bus is simply the number of signals on the bus that are dedicated to transferring information. It is usually expressed in bit transfer capability, i.e., 8-bit, 16-bit or 32-bit wide bus. Ideally, the width of the bus should equal the width of the internal processor data word. Otherwise, buffering or multiplexing is necessary to compensate for the different data word size. With most of the new high-performance processors like the 68030, the 80386 and the MicroVAX offering a 32-bit word size on the processor, it is much more common to see 32-bit wide buses. One of the many differences between the MicroVAX I and II is the width of the bus going from the 16-bit wide Q-bus to a 32-bit wide local memory bus.

Clock Speed

Many computers have a system-wide clock that sends out high-frequency "ticks" (or cycles) by which all internal system events are coordinated, including transactions occurring on the bus. The clock frequency is expressed in megaHertz (MHz), each frequency unit defining one clock signal cycle. Clock speed is also dependent on the physical length of the bus, because of noise and transmission line requirements. Typical clock speeds range from an IBM PC's slow 4.7 MHz (or 4.7 million cycles/second) to 33 MHz in a VAX 8650.

Many system designers specify their system clocks to run well beyond the capabilities of current hardware to facilitate future growth and upward compatibility. In asynchronous bus machines (see next section for explanation), clocks may run at different speeds in different parts



System Bus—16 Bits Wide

of the machine. This technique allows individual hardware devices to benefit from faster speeds as their individual design requirements permit. However, the limitations of other slower hardware may prevent the speed improvement from being fully reflected in the overall system performance.

Designing a bus for extremely fast clock rates takes attention to details such as signal transmission theory and adequate signal termination. If a bus is well designed, it should be able to improve as fast as the processors can. Many upgraded systems, such as the VAX 8650 upgrade from the 8600, amount to little more than running at a faster clock speed. If all the devices and the bus are capable of running with a faster clock, this is no problem. In some low-end systems, such as the PDP 11/73 and 11/83, customers simply changed the crystal on their systems to achieve substantial improvements in performance.

Bus Bandwidth

Bus bandwidth refers to a bus's maximum capacity for transmitting data. In simple terms, it is analogous to determining the volume of water through a pipe if you know the speed of the water and the pipe's diameter. Bus bandwidth is determined by its data transfer cycle time (x bytes/nanosecond) and its width (16 bits or 32 bits), and it is expressed as total data transferred per second (i.e., 1/2 kilobyte/second). The significance of a bus's bandwidth is that a bus with comparatively slow cycle time but with a wide bus width (or vice versa) can still have a competitive data transfer rate. For example, the Micro-VAX II has a wide 32-bit memory bus and a relatively slow 400 ns memory cycle time, but still is fast enough to keep up with the processor.

Bus Protocols

Synchronous Bus

This term refers to a timing method for synchronizing the transmission of data over the bus via regular signals determined by the system clock. The clock signal determines the precise and regularly occurring moment that the bus hardware will recognize and act on a signal level change (see timing diagram 1 at right). In a synchronous bus, the clock speed is perhaps the single most important determinant of bus performance.

Asynchronous Bus

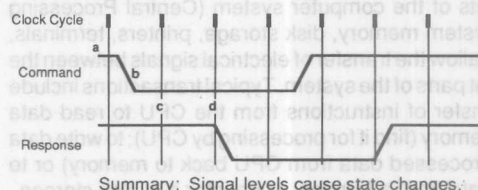
In contrast, an asynchronous bus has no system-wide clock. The bus hardware can recognize and act on a signal state change the moment it occurs (see diagram 2). Whether a bus should be synchronous or asynchronous is often hotly debated, each viewpoint with valid points to offer. Much of the debate focuses on the relative

merits of the Multibus II (synchronous) vs. the VMEbus (asynchronous); however, the arguments are relevant to all buses.

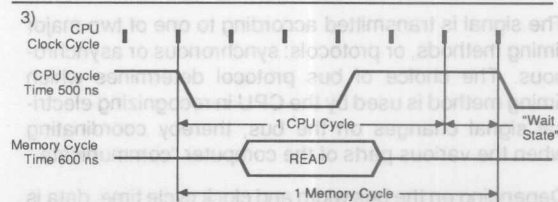
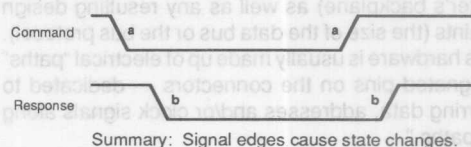
The synchronous buses (like the Multibus II and the BI) are generally considered to yield higher performance than the asynchronous buses. In general, the design criteria for a "handshaking scheme," acknowledging that a device is ready to transfer data is greatly simplified in a synchronous environment. On the other hand, a disadvantage of synchronous buses is that a change in a signal cannot be recognized until the next clock edge (see diagram 3). A small increase in the speed of another device on the bus will have no overall system performance benefit unless it is at least one full clock cycle faster.

There are other pluses of a synchronous bus. One reason is that it is easier to implement concurrent cycles (several cycles taking place over the bus concurrently) on a synchronous bus. Likewise, the greater degree of control enforced by the timing specifications generally results in fewer compatibility problems down the road.

- 1) Synchronous Bus - (a) At leading edge of each clock signal, other signal paths are sampled for a state change (i.e. high to low). (b) A signal change that occurs during a clock cycle cannot be sampled until (c) the next clock cycle. (d) The resulting signal change (response) caused by the command signal is implemented at the next clock cycle.



- 2) Asynchronous Bus - There is no system clock. (a) Upon a change in the command signal, (b) the response signal reflects a state change.



Synchronous and Asynchronous Protocols

Asynchronous buses are advantageous for other reasons. First, there is no waiting for clock edges before signal changes are acknowledged on the bus. As a result it is easier to take advantage of faster memory chips. The difference between the time when a device on the bus is ready to respond and the time a clock cycle strobes that response is called "clock latency." Since asynchronous buses do not have clock latency, there can be a real advantage to speeding up access times.

Other disadvantages include the increased possibility of design complexity resulting in performance inefficiencies and design errors. An asynchronous protocol is only beneficial if the designer can effectively mix different cycle times to enhance performance. In addition, a bus that is too complex may cause compatibility problems down the road. Another disadvantage of an asynchronous bus occurs when devices that plug into the system are synchronous (i.e. processors, peripheral interfaces, input/output devices). The bus signals must be synchronized to the local device clock, resulting in an additional time-consuming layer of activity and slowed performance.

Memory Processing Performance

Bus efficiency and microprocessor speeds are critical to system performance, but memory plays a significant role also. Its design is just as critical to overall performance.

Read Access Time — The most common measure of memory board performance is access time. Generically, access time is defined as the time from when the processor or other device makes a request for data at a given address to the time the memory board responds that it is ready to send data. This definition is subject to a fair amount of vendor variation. DEC, for example, defines access time on the Q-Bus in such a way that many vendors offer memory boards with access times faster than 100 ns. This is despite the fact that the memory

chips used on those boards have access times of 150 ns.

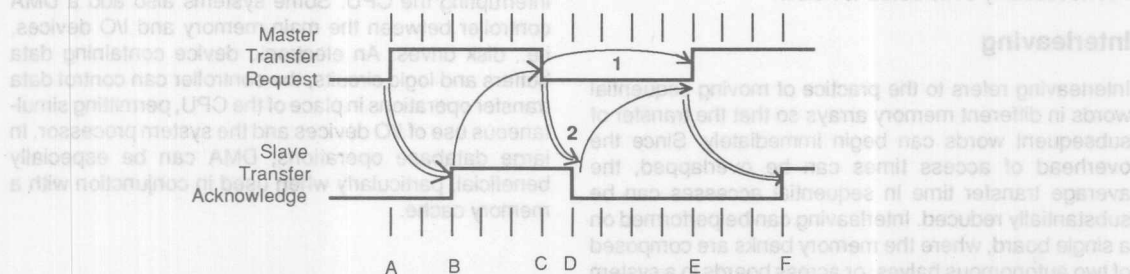
This paradox is a definitional quirk; the boards send the response signal (TRPLY) to the processor in advance of the data actually being ready. Since by the time the processor or other device is really ready to receive data, the memory board will be ready to send, this is allowable. The statistic is meaningful for comparison purposes with other Q-Bus memories, but is totally meaningless for comparisons across buses. In addition, it has little to do with performance on the Q-Bus. Almost all memories with access times less than 150 ns perform nearly identically on the Q-Bus.

Write access time is defined as the time from when a device sends data to memory until the next time it can send data. From the memory board's perspective, the data must be latched and a signal sent back on the bus indicating the data has been received. Since this is very straightforward, most write access times are very fast and do not vary considerably among vendors.

Cycle Time — The time from when a device makes a request for data until the next time a request for data will be acted upon by the memory board is defined as the read cycle time. Frequently cycle time is a better indicator of actual memory board performance than access time, especially in block transfers where the memory board is likely to be the constraining device.

In many cases design engineers optimize for access time and the result is a very slow cycle time. Which is more important depends on the application: I/O-intensive applications with lots of sequential reading and writing need a fast cycle time, while processor intensive tasks with highly random reads and writes depend more on access time.

Write cycle time is similarly defined as the time from a request to send data until the next time the memory board is able to write data.



The timing diagram above illustrates both access time and cycle time. Access time is defined as the slave receiving all signals at A until the slave sends back a response at B. Cycle time is from B to F including the full time of the subsequent access from E to F. Path 1 and path 2 indicate two popular "handshaking" schemes for master and slave devices.

Read-Modify-Write—The time from a read request until the memory board has latched the modified data and released the bus, including the time the processor needs to perform the operation, is defined as read-modify-write. Since this is highly processor-dependent, it is not often used as an indicator of memory board performance per se. It is a useful indicator for system-level comparisons.

MEMORY DESIGN FOR IMPROVED SYSTEM PERFORMANCE

BUS EFFICIENCY

Over the years, techniques have evolved that increase bus efficiency, i.e., decrease the incidence of bus transactions encountering wait states. These techniques show up over and over, in combinations or individually. It is important to understand the principles by which they work and interact with each other.

Multiplexing Data and Address

Multiplexing, or muxing as it is frequently abbreviated, is the alternate use of the same bus signal lines for data and address. The purpose is cost reduction: by sharing lines, the total number of signal lines is reduced by the lesser of the number of address or data bits. On the Q-bus, for example, 22 address bits and 16 data bits are muxed on 22 signal lines. The overhead of muxing can be considerable: additional signal lines are needed to enable the coordination or handshaking to inform devices what type of information is currently on the bus. Typically, multiplexing entails a performance penalty because the same lines have to perform two jobs. In transferring information the address must first be transmitted, followed by the data. If more than one consecutive piece of data must be transferred, this method quickly becomes inefficient, although block mode transfers can compensate to some degree. Multiplexing is now widely used on very high-performance buses like the Multibus II and the BI, so it is not necessarily considered too slow.

Interleaving

Interleaving refers to the practice of moving sequential words in different memory arrays so that the transfer of subsequent words can begin immediately. Since the overhead of access times can be overlapped, the average transfer time in sequential accesses can be substantially reduced. Interleaving can be performed on a single board, where the memory banks are composed of two autonomous halves; or across boards in a system or array. Many different interleaving schemes are currently in use, from the two-way interleaving on the VAX 780 and IBM RT PC to 8-way interleaving on some Data General systems. However, increased interleaving does not result in a linear increase in performance.

Transfer Methods

More efficient transfer of data and instructions can speed up system performance significantly. Since most transfers involve sequential addresses, methods for moving consecutive words or blocks at one time are particularly beneficial.

Prefetching and Pipelining

Prefetching refers to the CPU's ability to anticipate data accesses and start data retrieval before it is requested. A common form of prefetching is to start two to eight accesses in parallel so that the second (to the *n*th) access is proceeding simultaneously with the period where the first address is valid. The first access is subject to normal access times (150-300 ns) while subsequent accesses appear to be 10-20 ns apart. Hence, it is possible to transfer eight words in the time it would normally take to transfer two.

Pipelining is similar to prefetching, but it usually prefetches instructions rather than data.

Page and Block Mode Transfers

Both of these methods of transfer pump large bursts of data to and from sequential addresses. Instead of saving up a series of sequential addresses which will then be transferred consecutively, the CPU gives one instruction with a beginning address and the transfer takes that address plus the following page or block of addresses during the transfer. This technique is particularly useful in cases where the system is accessing the disk and multiple consecutive disk accesses would dramatically impair system performance.

Direct Memory Access (DMA)

DMA, an architectural feature of most buses, allows information to be read from disk and written to memory (or vice versa), via dedicated bus signal paths without interrupting the CPU. Some systems also add a DMA controller between the main memory and I/O devices, i.e., disk drives. An electronic device containing data buffers and logic circuits, the controller can control data transfer operations in place of the CPU, permitting simultaneous use of I/O devices and the system processor. In large database operations, DMA can be especially beneficial, particularly when used in conjunction with a memory cache.

Caching Memory

Caches have become one of the most widely used techniques to improve the performance of systems in general. Caches can be located on the processor, the memory or the peripherals (such as a disk drive), and serve as quickly accessible storage for interim processing results, soon-to-be-executed instructions or blocks of sequential addresses.

A cache memory on the processor is defined as a small (usually 16 KB to 64 KB byte capacity), very high speed (50 ns access time or less) memory that is tightly coupled with the processor. It is usually implemented in Static RAM or ECL to achieve faster access times. The use algorithm, unlike main memory, is implemented in firmware rather than in the operating system. The use algorithm is selected to achieve the maximum cache hit rate without tremendous overhead in "swapping." The optimal size of a cache is really an economic decision: it is driven by the difference in cost between main memory and cache memory. The larger the cache gets, the less likely it is to see enough performance increase to justify the dollar cost of additional cache memory. Hence, caches are generally quite small.

Caches on the disk controller are currently popular, now that a megabyte of memory can be condensed into one or two square inches. Implemented in DRAM, they are really just an extension of main memory with a different use algorithm. In sequential DMA, a cached disk controller can achieve extremely fast transfer rates.

Local Memories

Local memories are being used principally because of multiprocessing. In multiprocessing systems, there are often two or more processors operating concurrently from one memory. While the multiple processors can provide significant "number-crunching" functionality, the

performance enhancement can be diminished as the system bus is overloaded by processor-memory transfers. To mitigate this problem, most multiprocessor architectures allow for each processor to have its own local memory on a separate bus.

Some systems completely eliminate system memory in favor of local memories. The difficulty in maintaining local and system memories has to do with control of processes, synchronization and concurrency. The benefits of better performance have made the solution of these problems a necessity for multiprocessing operating systems.

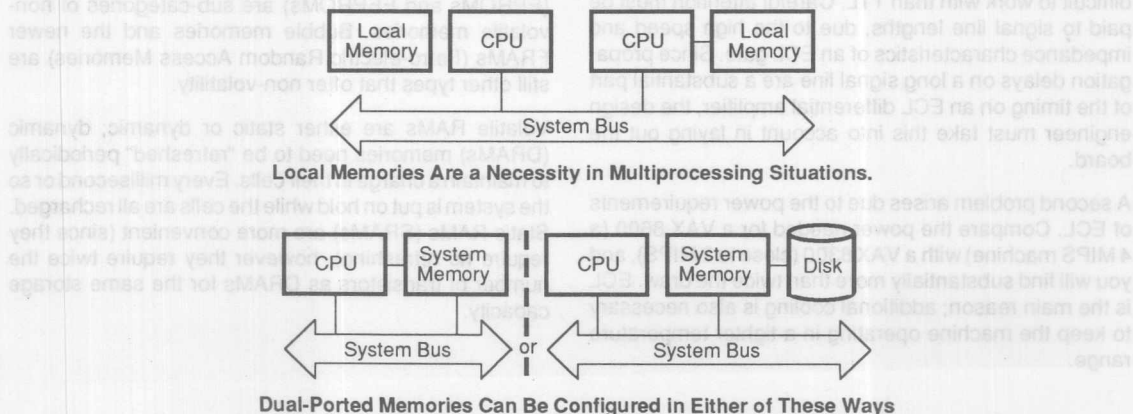
Dual-Ported Memories

Dual-ported memories are similar in many ways to local memories except that they can answer on either a local bus or the system bus. The advantage of a dual-ported memory is that it both increases bus efficiency by taking traffic off the system bus, and it has a faster access time than transfers on the system bus.

Dual-ported memory is typically situated on both the system bus and a local bus to the processor. However, there are dual-ported memories that sit on a local bus to a disk controller or other high-speed peripheral like an array processor. The principle of operation is essentially the same where ever it sits. Dual-ported memories are generally much more expensive than standard single-ported memory because of the considerable additional logic required.

Array versus On-Board Memory Control

One of the most common features of today's high performance systems, from the DEC VAX 8800 down to the IBM RT PC, is the use of separate memory controllers and array cards. The memory controller is that portion of a memory that performs addressing, timing, refresh



control and arbitration, and EDC or parity generation and checking. By locating the memory controller on a separate card from the memory array, or simply directly on the processor card, expansion and manufacture of memory cards is greatly simplified. In addition, with the rapid advance in memory chip density, memory controllers are not obsoleted along with the array cards. The big advantage of separating the memory controller from the array is that the private memory bus can be greatly simplified from the full system bus. Typically it is much shorter; hence, the problems with noise are substantially reduced.

The second big advantage is cost: the memory controller accounts for one-third to one-half the cost of a memory card (with an on-board memory controller). When large numbers of arrays are required, eliminating the repetition of the memory controller can amount to considerable savings. While some recent buses still use on-board memory controllers (such as the VAXBI), the trend in system design is toward separating the controller from the array. The Sun 3/1XX series, for example, uses the high performance VMEbus for a system bus, but the memory hangs on a private memory bus with the memory controller on the processor card. Similar designs are also used in the RT PC and the VAX 8500, indicating the wide range of system sizes that are going in this direction.

ECL vs. TTL

Emitter Coupled Logic (ECL) is not a new idea — many RCA and CDC systems from the sixties pioneered the use of ECL for achieving better performance. ECL eliminates transistor storage time as a speed limiting characteristic, permitting much higher performance than is possible with TTL (transistor-transistor logic) circuits. ECL is now being used by DEC in the VAX 8600, 8700 and 8800 series; this has re-ignited interest by other manufacturers as well.

There are several problems with ECL that make it more difficult to work with than TTL. Careful attention must be paid to signal line lengths, due to the high speed and impedance characteristics of an ECL gate. Since propagation delays on a long signal line are a substantial part of the timing on an ECL differential amplifier, the design engineer must take this into account in laying out the board.

A second problem arises due to the power requirements of ECL. Compare the power needed for a VAX 8600 (a 4 MIPS machine) with a VAX 8300 (close to 2 MIPS), and you will find substantially more than twice the draw. ECL is the main reason; additional cooling is also necessary to keep the machine operating in a tighter temperature range.

ECL is generally only used in the busiest circuits. To achieve high densities in the system memory, standard dynamic RAMs are the technology of choice. ECL RAMs are too expensive and too hot to use for the large 8–128 MB arrays that are now commonly available for system memories. Instead, the memory controller is designed using ECL, and ECL-to-TTL conversion and buffering are performed on the arrays.

MEMORY TECHNOLOGIES AND HIGH PERFORMANCE

Amid the hoopla of high performance microprocessors and “war of the buses,” the advances in memory technology often are ignored. In fact, memory has come a long way since the core memories of the sixties. Semiconductor technology averages a fourfold increase in density approximately every 2–3 years. These improvements in densities and processes often pave the way for the microprocessors to follow. This section gives a brief review of the memory technologies currently in use, those in process and what that means for high-performance computers.

What Is Memory?

Memory is that part of the computer system from which the CPU reads and writes information which it uses in the execution of programs. Memory technologies are characterized by volatility, write-ability, and semiconductor technology. Non-volatile memory maintains its storage after power has been turned off. Write-ability is the extent to which data can be changed, and the means used to change data at a given location. Semiconductor technology refers to the actual semiconductor and substrate used for manufacturing the memory device.

Non-volatile memories are generally used for programs or data that must remain intact through power and system failures. Read-Only Memories (ROM), Erasable, and Electronically Erasable Programmable ROMs (EPROMs and EEPROMs) are sub-categories of non-volatile memories. Bubble memories and the newer FRAMs (Ferro-electric Random Access Memories) are still other types that offer non-volatility.

Volatile RAMs are either static or dynamic; dynamic (DRAMs) memories need to be “refreshed” periodically to maintain a charge in their cells. Every millisecond or so the system is put on hold while the cells are all recharged. Static RAMs (SRAMs) are more convenient (since they require no refreshing), however they require twice the number of transistors as DRAMs for the same storage capacity.

Within volatile RAMs, the major semiconductor technologies in use include N-channel Metal Oxide Semiconductor (NMOS), Complementary Metal Oxide Semiconductor (CMOS), Emitter-Coupled Logic (ECL), and Gallium-Arsenide RAMs.

GaAs RAMs are not currently widely available, although the military is funding considerable research in this area. ECL SRAMs are only used in cache memories because of their high speed and very high cost. NMOS has been the principal DRAM technology for many years, and is now being replaced at the high end by advances in CMOS technology. Megabit DRAMs are now almost exclusively CMOS.

CMOS is both faster and uses less power than NMOS. With density and cost approaching NMOS levels, CMOS is fast becoming the technology of choice.

The Meaning of Specifications — Most vendors quote typical, average or maximum specifications for their boards. "Typical" is supposed to be the speed for an operation characteristic of normal usage. "Maximum" is defined as the most it will ever be in any operation. "Average" is sometimes used synonymously with "typical" and sometimes to reflect an average of buffered and non-buffered operations. This is relevant when some operations will occur out of a buffer on the memory board rather than through the memory chips. What is truly "average" of course depends on what a normal mix of buffered and non-buffered accesses should be. Maximum times are supposed to be based on the worst-case timings of all the devices on a board. Since this number may be impossibly long, most vendors simply calculate a number they are confident will always be greater than actual usage.

Changes in the specified speed of a board do not always translate into actual improvement in system performance. This is because the memory board may not be the bottleneck in your system. If another area of the system is the binding constraint, speeding up the memory card will have little additional value. It may still be worthwhile to select the faster card, however, because over the life of the system it is likely that faster hardware will be added.

In a number of cases, the advertising of faster memory is truly misleading. The Q-Bus is a good example. Vendors achieve very fast access times by a variety of means, some in clear violation of the Q-Bus specifications. (Hanging logic directly on the bus without the appropriate drivers is not permitted, for example.) Regardless of how fast the access time is, if the processor or other device is not ready to receive data, the reduction in time is meaningless. This is especially true if cycle time was sacrificed to achieve these spectacularly low numbers.

Another example is the VAX 8600. Faster memory on the 8600 is feasible by definition, but it has little to do with actual system performance. Some vendors who advertise memory faster than DEC's "prove their case" by asking customers to switch off the cache on the processor. Some customers are actually impressed that their memory could perform faster if their system were non-functional.

In general, cached processors have done a lot to simplify the world of the memory vendor. When only 20% or 30% of the accesses are to main memory, the speed of memory is rarely the most important attribute.

Techniques Used to Improve Performance

The simplest way to speed up access time on a memory board is to use faster DRAMs. Most 64K and 256K DRAMs on boards today have a 150 ns access time. For a premium, there are good quantities of 100 and 120 ns parts available. The megabit DRAMs are yielding even faster parts; 100 and 120 ns access times are more common than slower devices. As die sizes shrink, the smaller circuits become faster and faster.

The access time of the memory chip is by far the largest component of the access time on the memory board. Hence the largest proportional gains in speed are achieved at this level. To achieve a gain of 30-50 ns through better design optimization and faster memory controller logic is very difficult.

At the memory controller level, faster memories are achieved through a variety of techniques. By focusing on the longest chain of logical gates, the design engineer attempts to whittle this down to the absolute minimum. He then can try using faster logic like CMOS (Complementary Metal Oxide Semiconductor) or ALS (Advanced Low-power Schottky) that pare each gate to the minimum interval. The trade-off at this level is between more expensive, faster parts and less-expensive, older, commodity-type logic.

Beyond this, there are various ways to "cheat" on timing, some of which are innocuous and some of which play with the ultimate reliability of the circuit under worst case environmental conditions. On the Q-Bus, for example, the bus is specified to be functional with backplanes up to 50 feet long. Since this is a rarity, to say the least, some designers have been known to make assumptions that the actual maximum is somewhat less than that. Also common is ignoring DEC requirements for specific drivers on all bus interfaces. If it works, many customers don't particularly care.

Another way to save time is by cutting margin to the bone. All devices have worst-case and actual performance

specifications. By skimping on worst-case timings, most circuits can achieve considerable improvements. If the circuit never fails, is this imprudent? The problem comes when the circuit only fails rarely or under adverse environmental conditions.

To some degree the customer is dependent upon the manufacturer for prudent design decisions. There is no substitute for adequate testing and design verification at the alpha and beta test stage to reveal a sound design. Testing at elevated temperatures (55–60°C) is also a good preventative measure, since device timings are at their slowest at higher temperatures. High temperature failures are the most common way to find out if an engineer "pushed the envelope" a bit too far.

Other techniques that are sometimes used on board level products are borrowed from system level techniques: interleaving and caching. On-board interleaving is sometimes implemented on a single card even when it is not specified in the system architecture. While the benefits of interleaving are not as great over a system bus, it can still boost performance.

Caching or buffering is another popular approach. Digital has implemented a two-stage cache on the MicroVAX 3X00; 1 Kb of 90 ns cycle on-chip cache, and 64 Kb of 180 ns cycle on-board cache. Intel has implemented memory board-based caching in their Multibus II product line.

Clearpoint uses a 64-bit buffer in an EDC chip set that improves access times on sequential reads and writes. The basic idea is to have a modest-sized buffer on the memory card that latches consecutive addresses each time data is accessed. Then if the subsequent operation calls for consecutive data, no additional access to the DRAMs is necessary. Since the buffer can usually be accessed in one-half to one-third of the time of a DRAM access, this considerably improves performance. Providing a large cache on the memory in addition to a cached processor, however, is not likely to yield incremental results.

Density

After performance, density is the most important feature of a memory card. Customers always want more memory in fewer slots with less power consumption. Greater density reduces the cost of additional memory, since the last megabyte on a board is always a good deal less expensive than the first. What follows is a compendium of techniques used to increase density.

DRAM Capacity

The most basic means of increasing density is to use DRAMs with a higher capacity. The DRAM manufacturers have obliged by coming up with denser memory chips

year after year. Currently, a new generation of memory boards is entering the market based upon megabit DRAMs. This will supersede the 256K DRAMs that are now the bread and butter, just as the 256K DRAM replaced most of the 64K DRAM product. The inevitability of this progress is now old hat. The only issue is when the crossover occurs.

Typically sales begin on the highest density product as soon as a product can be delivered. With the 256Kb DRAMs this was practically instantaneous since the device is pin-compatible with the 64Kb DRAM. The megabit DRAM took longer since it is an 18-pin device instead of 16-pin like the 256Kb and 64Kb DRAMs. Demand shifts slowly, typically crossing over when the 4x density device costs 6-7 times as much. The crossover occurs before the price has decreased to 4x because the denser parts obviate 3 PC boards full of interface logic for the same density. In addition they are more reliable (because of fewer parts), use less power and free up slots.

Curiously, demand for the less dense parts continues for years after they are non-economic for design-ins. 64K DRAMs are still widely used even though they now cost more than 1/2 as much as 256K DRAMs. Once a part becomes more of a specialty item than a commodity, the price begins to rise.

DRAM Packaging

DRAMs are available in a variety of packages, each of which allows for a different packaging density. The most common package by far is still the Dual In-line Pinpackage (DIP), which looks like a standard IC with a row of pins emanating from each lengthwise edge. The popularity of this package is primarily the result of history — most computer products are still designed around DIPs, on standard printed circuit boards using through-hole technology, and soldered over a wave solder machine. DIPs are easy to handle and insert, their height is minimal, and they have been widely used in the past.

Surface Mount Devices (SMDs) offer much greater packaging density than DIPs and have been growing in popularity for this reason. Originally developed and popularized in the Far East for applications in toys and small appliances, SMDs require less than one-half the surface area of a DIP for the same capacity. In addition, the leads do not penetrate the PC board so that devices can be mounted on both sides of a card if the system will allow this. The leads are folded so that they lay flat either underneath the device or to the side of it.

Manufacturing of SMD-based boards requires a whole new set of equipment very different from that used in conventional assembly. Because the devices are so much smaller and the placement on the board is not

guided by insertion into holes, automatic assembly is preferred. The "pick-and-place" machines that are used for this are very sophisticated, incorporating the latest in robotics technology. SMDs are glued into place using a solder paste; instead of being waved, the boards are passed through a hot vapor chamber that melts the solder and makes the connections.

Eventually, SMDs should pass DIPs and conventional packaging in popularity, once all the manufacturing wrinkles are ironed out and the cost comes down. Right now, it is still a relatively expensive package and the assembly costs are still greater than manual assembly. Because of the tight tolerances, only small PC boards are currently used with SMD assembly. On larger cards, such as DEC's 16 MB 8600 memory, DEC chose to use SMD technology on the small daughter cards rather than attempt their use on the full-size PC board. Clearpoint uses SMDs on boards as large as the VAXBI form factor, as well as on the smaller cards for the MicroVAX 2000, Apollo DN 4000 and daughter card for the VAX 8800.

SMDs have spawned another package that has some of the advantages of SMD without any penalty except cost: Single In-line Pin packages or Single In-line Memory Modules (SIPs and SIMMs). SIPs are essentially small PC cards with a row of SMDs mounted on one or both sides, with normal insertion pins along one edge of the PC card. The SIP is then mounted on edge on a standard board with the pins penetrating the PC card just like any other conventional IC. With SIPs it is possible to achieve densities up to three to four times what is possible with DIPs; however, the height is sometimes greater than any other device on the card. In systems that allow for wide spacing among cards, this is not a problem. In most small systems these days, however, it is a concern.

Another issue is cost. When SIPs were first introduced, the military was the principal customer and prices were several times comparable product in DIPs. Now that the market has expanded and several of the major DRAM vendors have entered, the price is lower, but still 50% to 75% greater than DIPs.

The latest package to hit the market is ZIPs — Zig-zag In-line Pin Packages. ZIPs essentially take a normal DIP package and stand it on edge, with both rows of pins emanating from the same edge in a zig-zag pattern. ZIPs have a lower profile than SIPs, but still allow for twice the density of DIPs. The manufacturing cost is nearly identical to DIPs, however the pricing reflects the smaller market size since it is a new product.

Mitsubishi introduced the product in 1985 to little initial interest. The MicroVAX II created a niche for the product because ZIPs were the most cost-effective way to pack 8 MB of memory on a single card. Suddenly, demand

soared and now most DRAM vendors are offering the package. As cost settles in to the DIP range, the ZIP market should continue to expand. The only difficulty seems to be keeping the devices in place as they pass over the wave-solder machine.

In the long run, it is difficult to tell whether SMDs will win or whether advances like ZIPs will extend the life of conventional technology. An often overlooked fact is that the wafers are now becoming a much larger fraction of the size of the package. Megabit DRAM SMDs are not much smaller than DIPs.

NEW TECHNOLOGY

A number of other developments have made greater densities possible. Custom and semi-custom Very Large Scale Integration (VLSI) devices are now popping up in many applications. VLSI has declined in cost substantially over the last two to four years. CAD/CAE tools now allow the rapid design and simulation of very large circuits. Clearpoint, for example, has a two-chip set for EDC control, comprised of 2400 and 5200 gates per device. The result is that what used to take 50 - 100 ICs to implement in standard logic now can be done with one or two VLSI chips. The space freed-up on the card can now be used for even more memory chips, or some additional functionality.

Some prognosticators are forecasting that FRAMs (pronounced F-RAM) will obsolete DRAMs. FRAM stands for Ferroelectric Random Access Memory. It is a new memory technology, based on an old discovery dating back to 1921. Promising significant enhancements over standard DRAMs — such as non-volatility, radiation-hardness and density — it could be the perfect memory technology.

The ferroelectric effect specifically refers to the tendency for certain crystalline materials to polarize spontaneously when an electrical field is applied, *and to remain polarized after the field is removed*. If the electrical field is reversed, the polarization is also reversed. The result is that the crystalline material can act as a capacitor with two distinct polarizations dependent on voltage levels. Since no current is required for the ferroelectric material to retain its polarization, it can act as a completely non-volatile digital memory capacitor. Storing either 1's or 0's in a ferroelectric element, the FRAM can be read by sensing the interaction of an applied field with the element's polarization.

Recently, two companies have announced breakthroughs in materials and processing necessary for commercially viable FRAMs: Ramtron Corp. (Colorado Springs, CO) and Krysalis Corp. (Albuquerque, NM). Ramtron currently produces a 256-bit non-volatile static

RAM which has a thin film of lead zirconate titanate (PZT) deposited over conventional semiconductor memory circuitry to form ferroelectric capacitors as part of the memory cells. The ferroelectric capacitors only come into play in the event of a power interruption. Krysalis likewise uses a conventional CMOS silicon wafer over which a thin film of ceramic ferroelectric material is deposited. The film is delineated to form non-volatile memory cells. In contrast to Ramtron's process, Krysalis' ferroelectric cells will serve as the primary storage element.

Both companies are still in early stages of development and promise much in years to come. Already a possible replacement for EEPROMs, they could eventually supplant SRAMs and DRAMs if they deliver the promised speeds and longevity. But don't hold your breath. FRAMs aren't forecast for widespread distribution until 1992.

Another development is the widespread use of Programmable Logic Arrays (PLAs or PALs), PROMs and EPROMs in situations that formerly used conventional logic. As hardware and software expertise merges, so do the approaches to engineering solutions. Programmable devices allow for very dense packaging combined with incredible flexibility. It is not uncommon today for EPROMs or PLAs to serve as upgrades to a board to enable compatibility with the latest developments on a bus. Whereas previously boards might be returned for extensive ECOs and retesting, replacing a socketed PROM in the field is a simple and inexpensive procedure.

DESIGN INTEGRITY

Engineers refer to design integrity wistfully as the "right stuff": the invisible glue that brings coherence and consistency to any system or product. Most seasoned engineers sit back and tell you it's like a work of art, "you know it when you see it." For the layperson, a few pointers would be helpful.

Is It the Solution to the Right Problem?

Any design or system is an attempt to solve a problem or a set of problems simultaneously. It should be very obvious when looking at a design "solution" what the problem is. Frequently, however, a design is borrowed from another setting and haphazardly applied to the given problem. If it works, little attempt at modification or optimization is made.

Examples abound in the computer world. The Q-Bus was originally a stop-gap inexpensive microcomputer bus intended for very low-end applications like process control and single-user systems. It is now the system bus on the MicroVAX 3X00. While it may be expedient for DEC to maintain a performance gap between its high- and low-end systems, the continued tweaking of the Q-

Bus to apply it to a high-performance environment leaves a lot to be desired from the customer's point of view.

Was It Designed to Grow in Predictable Ways?

A design should be dynamic; it should be adaptable to the changing environment that is inevitable in most computer systems. Many of these changes are predictable, such as the continuing increase in memory chip densities. Does the design plan for growth? Are constraints built in because of compromises that will undoubtedly become bottlenecks in the future?

Are There Hints of Capabilities Well Beyond Those Required?

This is a corollary to the above question. Usually a growth path will be revealed by huge margins or extra capabilities on a board that are not warranted by the current environment. If these extras are costless or nearly so, they hint at the forward possibilities a customer should expect.

Other indications are the use of advanced technologies that solve old problems in different ways than normally done. VLSI in a unique application, a novel test process or reliability feature, all are evidence of original thought beyond what is called for in a design. It is also evidence of the vendor's capability to solve more difficult problems as they arise in the future.

Is It a Clean Design?

There is no substitute for the "right stuff" — a well-conceived design or an intelligently packaged system shows the attention to detail that is necessary in today's market. Compare several designs and look for the differences.

If Compromises or Trade-offs Are Made, Do They Make Sense?

Often compromises are made to solve one problem, creating new problems or costing too much somewhere else. Look for the details that don't make sense: special instructions where there should not be any; inconsistent performance measurements that are optimized for unusual environments; advertising key features of a product with no mention of those attributes that are known to be trade-offs of the feature. How many times, however, have you bought something that sounded too good to be true only to find out the "gotchas" when you took delivery?

Designed for Reliability: Keep It Simple

One of the most common phenomena in today's market is a very advanced design that never seems to work. A good design should show evidence of addressing reli-

ability as a central problem. How is the system likely to fail? Where are the design decisions that address these failure modes? Is it easy to test and stress these failure potentials in the quality control process?

What Happens When You Scratch the Surface?

Usually the design represents a coalescence of many competing concerns and constraints. In many ways, it is an optimization problem in many dimensions. The result is, when you scratch the surface (or formally, do sensitivity analysis on each of the variables) you expect to be forced back to the optimization point reached by the design engineer. Needless to say, sometimes this does not happen. Even a layperson can ask questions that delve into these design decisions, revealing the layers of the problem below. If you find questions that are not asked, and choices that seem to have been decided randomly, you are not looking at "the right stuff."

The High Performance Design

The result of a careful design is a product that stands the test of time. Understanding the design decisions for high performance put you, the buyer, in the driver's seat, right where you belong.

Reliability is of paramount concern to all customers. The lack of consistently reliable product has spawned a growth industry within the computer market for highly reliable systems, such as Tandem and Status. Since many decisions on the system level have important ramifications for reliability at the memory board level, this chapter is devoted to clarifying the issue.

THE PROBABILITY OF ERRORS

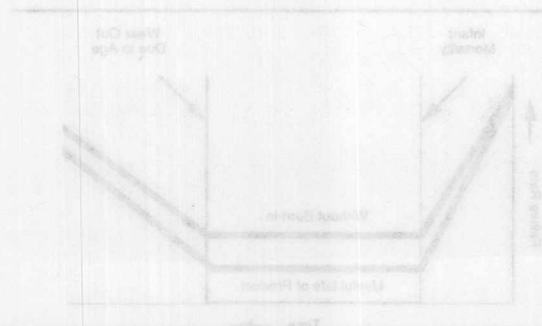
There are two types of memory errors in dynamic RAMs: soft errors due to radiation-induced bit switching, and hard errors due to the unexpected deterioration of a memory chip. Soft errors cause no lasting damage to the memory board, but they do corrupt programs or data. Hard errors demand physical repairs. The principal cause of soft errors is alpha particle radiation from trace levels of uranium and thorium in the plastic packaging of a DRAM. Other cosmic sources of high-energy radiation also can cause soft errors. The energy necessary to displace electrons from an individual memory cell is a function of the size of the cell and the charge it carries.

As the cell size decreases by a factor of four, as in the change from a 64Kb to a 256Kb DRAM, the charge per cell only decreases slightly. The probability of a highly energetic particle hitting the device is a function of the size of the device. Since the 256Kb DRAM die is only slightly larger than the 64Kb die, the soft error rate only increases perhaps 50% with a 4X increase in density. The result is that a 256Kb DRAM is somewhat less reliable than a 64Kb on a per device basis, but more reliable on a per cell basis.

Hard failures occur when devices on the memory board fail. Memory chips have been known to fail partially or completely; all ICs in the bus interface logic also have some probability of failure. ICs in general exhibit a "bathtub" distribution for failures over time (see chart below), with a high initial failure rate while undetected IC defects are uncovered, then a low failure rate for a long time until it increases again because of wear-out.

The probability of memory errors has been estimated by memory chip manufacturers, EDG chip set manufacturers, and various users. The estimates vary, not surprisingly, depending on what the writer is trying to sell. Robert McEliece, a professor at Caltech, wrote in Science, "The American ('The Reliability of Computer Memories', January, 1982 262, 788-82) about memory reliability will be a major problem in the next five years for

System hardware reliability is achieved by including some amount of overhead for identifying when errors have occurred and either halting the system or providing some measure of fault tolerance. The method by which the system addresses error protection is usually consistent throughout the hardware and software. The chain is only as strong as its weakest link: if the hardware is designed to identify faulty errors and the software has no provision for interrupts, the parity checking is basically worthless.



Designing for Reliability

Reliability is of paramount concern to all customers. The lack of consistently reliable product has spawned a growth industry within the computer market for highly reliable systems, such as Tandem and Stratus. Since many decisions on the system level have important ramifications for reliability at the memory board level, this chapter is devoted to clarifying the issue.

THE PROBABILITY OF ERRORS

There are two types of memory errors in dynamic RAMs: soft errors due to radiation-induced bit switching, and hard errors due to the unexpected deterioration of a memory chip. Soft errors cause no lasting damage to the memory board, but they do corrupt programs or data. Hard errors demand physical repairs. The principal cause of soft errors is alpha particle radiation from trace levels of uranium and thorium in the plastic packaging of a DRAM. Other cosmic sources of high-energy radiation also can cause soft errors. The energy necessary to displace electrons from an individual memory cell is a function of the size of the cell and the charge it carries.

As the cell size decreases by a factor of four, as in the change from a 64Kb to a 256Kb DRAM, the charge per cell only decreases slightly. The probability of a highly energetic particle hitting the device is a function of the size of the device. Since the 256Kb DRAM die is only slightly larger than the 64Kb die, the soft error rate only increases perhaps 50% with a 4X increase in density. The result is that a 256Kb DRAM is somewhat less reliable than a 64Kb on a per device basis, but more reliable on a per cell basis.

Hard failures occur when devices on the memory board fail. Memory chips have been known to fail partially or completely; all ICs in the bus interface logic also have some probability of failure. ICs in general exhibit a "bathtub" distribution for failures over time (see chart below), with a high initial failure rate while undetected IC defects are uncovered, then a low failure rate for a long time until it increases again because of wear-out.

The probability of memory errors has been estimated by memory chip manufacturers, EDC chip set manufacturers, and various users. The estimates vary, not surprisingly, depending on what the writer is trying to sell. Robert McEliece, a professor at Caltech, wrote in *Scientific American* ("The Reliability of Computer Memories", January, 1985 252:1, 88-95) about memory reliability with no obvious predisposition. He used the figure for

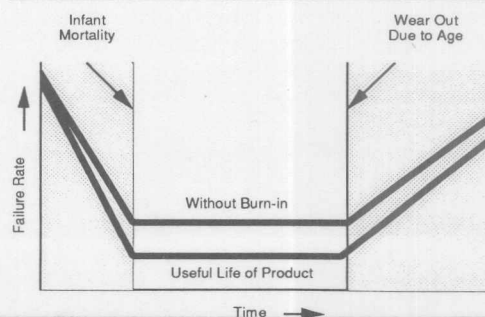
expected soft failure rate of a single memory cell of 1 every 1,000,000 years. In a 1 megabyte memory board with 8.4 million cells, this translates into a mean time before failure (MTBF) of 43 days.

Texas Instruments calculates the soft error rate MTBF based on their 64Kb DRAMs somewhat more optimistically; for an 8 megabyte system the MTBF was 33.4 days. Advanced Micro Devices, Inc., (AMD) a manufacturer of an EDC chip, estimates that a 16 MB system would have a MTBF of 13 days. Memory chip manufacturers measure reliability in terms of FITs—expected errors per billion device-hours. The 64Kb DRAM has a soft error rate of approximately 500 FIT, the 256Kb about 730, and the megabit DRAMs something close to 1000. Texas Instruments estimates hard errors to be roughly 1/5 to 1/3 as likely as soft errors.

Memory board manufacturers buy very few lots of memory chips in which the number of failures for 10,000 chips over a 72-hour burn-in is less than one; it is sometimes over five. While this represents the high initial failure of the early portion of this bathtub distribution of failures over time, it still implies a hard failure rate of about 1400 FIT (based on 1 per 10,000).

ERROR PROTECTION

System hardware reliability is achieved by including some amount of overhead for identifying when errors have occurred and either halting the system or providing some measure of fault tolerance. The method by which the system addresses error protection is usually consistent throughout the hardware and software. The chain is only as strong as the weakest link: if the hardware is designed to identify parity errors and the software has no provision for interrupts, the parity checking is basically worthless.



The three basic categories of error protection that are in use today are: parity generation and checking; error detection and correction (EDC); and complete redundancy. A fourth category — none of the above — is still seen occasionally. Parity checking is the most common low-end measure, since the cost is quite modest, performance is only mildly affected, and the software to support it is reasonably simple. EDC is the most common level of protection for minicomputers and mainframes, and seems to be migrating to both ends of the spectrum as hardware becomes less expensive and more reliable. Complete redundancy is the means used by "fault tolerant" systems that need to remain operational even when components of the system die. It is more than twice as expensive, since not only is twice as much hardware needed but the system software environment is considerably more complicated.

RUNNING BARE

The big question here is "Why?" Some OEMs still pay for reliable hardware and then fail to implement software that will take advantage of it. There is no more dismal feeling than having put in a week of work entering a database to find out that the system corrupted it days ago, and you've been copying bad files as back-ups. "Why didn't I know?" you may ask. Because your system was too cheap to put in any error protection.

"But I spent \$100,000 on this system!" There's one born every minute.

PARITY GENERATION AND CHECKING

Parity checking would at least eliminate the above problem. Parity checking involves storing a bit with every byte of information that indicates the internal consistency of that byte. Generally this is as simple as determining whether there is an odd number of ones in the byte, and storing a 1 in the parity bit if there is. Then, every time that byte is accessed, transferred, stored, etc., the parity bit is compared with the byte to make sure it is still consistent. If not, a parity error is generated, and the system is generally halted and the location of the error is backed out.

How does the system know where the parity error was found? Through the Control and Status Register (CSR). The memory controller has a register that latches or stores the address of the word currently being accessed. When the system halts, the CSR can be interrogated using basic machine language commands to reveal the row and column address. If the system is restarted and halts again at the same address, this usually means that a memory chip has died and needs to be replaced. Byte parity uses a parity bit for every byte of data. In a 1 megabyte memory board, there would be 128 64Kb DRAMs used to store data and 16 used to store parity bits. This, and some additional logic, are the

principal overhead of parity generation. Regardless of current prices, this is a very small price to pay.

ERROR DETECTION AND CORRECTION

Error detection and correction (EDC), also called ECC for error checking and correction, goes one step further by correcting single-bit errors that would otherwise halt the system. Double- and multiple-bit errors are also detected (unlike some multiple-bit parity errors) and treated like parity errors by the system.

EDC is valuable since the majority of errors are single-bit. EDC allows the continued operation of a system with only an entry on an error log to indicate that an error actually occurred. Hard errors due to memory chip failure are also corrected by the system, since each memory chip stores at most one bit from each word. Since hard errors are usually permanent, when the system is operating with a failed memory chip on board, it is essentially offering only parity protection for additional errors.

THE MECHANICS OF EDC

To the layperson, EDC seems almost too good to be true. Ask a casual observer to devise an error correction scheme, and he is likely to come up with two error-correction bits for every bit of data: store all bits three times; if one bit is different, go with the majority rule. This system requires three times as much storage as no protection, and gives incorrect information for double-bit errors. How, then, can an EDC system work that requires no more storage than parity memory (1 bit per byte), corrects all single-bit errors, and detects all double-bit and most multiple-bit errors?

EDC works by storing an error correction code (ECC) with each word that both identifies where a failure has occurred and corrects the error. The word size upon which error correction is performed depends on the specific memory design tradeoffs; typically, it is the word size of the system (usually 16 or 32 bits). But, it can use an ECC word size of 64 bits as well. Corrections can be made on either four 16 bit or two 32 bit words simultaneously.

In simple terms, the number of bits required to identify the location of an error in a word of N bits is $\log(2)$ of N. To understand why this is so, imagine trying to tell someone else where the error is with yes or no answers. They would ask "Is it in the first half?" then "Is it in the second quarter?" etc. Each time the possible locations would be halved.

Additionally, two bits are required no matter how many bits are in the word in order to allow for errors in the check bits and to correctly diagnose double- or multiple-bit errors.

While not delving too deeply into the mathematics, it is obvious that the $\log(2)N$ formula also has to include the check bits. Using the 64-bit word as an example, assume k check bits are required. Then $k \geq \log(2) (64+k)$.

Correctly diagnosing multiple-bit errors requires an additional bit. Operationally, only odd combinations of one's (1, 3, or 5) in the ECC are used to diagnose problems; even combinations indicate that double- or multiple-bit errors have occurred. If only one bit is a 1, this implies that the checkbit is in error. If 3 bits or 5 bits are one's, then the designated data bit is in error. If 2, 4 or 6 bits are one's, then there has been a double-bit error (or even multiple-bit error).

PARITY vs. EDC: A COMPARISON

The effect of EDC on reliability is substantial. Whereas with parity a single-bit error causes an interrupt, with EDC it takes two errors within a word to cause an interrupt. The MTBF due to two soft errors in a word is about 600 million years. McEliece estimates the MTBF of a one megabyte EDC board due to soft errors to be 63 years; for a parity memory the MTBF would be 35.7 days. There are, of course, more probable ways of crashing an EDC memory. Only the memory array is protected against hard errors. The failure of any of the interface ICs is uncorrectable. Even so, the MTBF of an EDC memory card is at least an order of magnitude greater than a parity memory; AMD estimates that a MTBF factor of 50 to 60 is expected.

The protection against hard errors is very dramatic: it is possible to pull a memory chip off the board and have the board continue to operate. This is because memory arrays are organized in such a way that each memory chip stores at most one data bit in a word. If one of the memory chips dies, it can only effect a single-bit error in a word, and is therefore correctable. (Note: this is only true with x1 DRAMs; 64K x 4 or 256K x 4 organizations can store more than one bit per word in a single DRAM.)

THE IMPORTANCE OF THE EDC WORD LENGTH

The chart above shows the number of check bits required for EDC compared to the number of parity bits required for byte parity checking. This corresponds directly to the number of memory chips that are required for a given capacity of memory. For example, an EDC 1 MB memory using a 32-bit word would require 3 more bits than parity on each row. Since there are 4 rows of 32 bits (using 64K DRAMs), this amounts to 12 extra memory chips required: 156 compared to 144 for parity.

However, the same number of memory chips are required for parity memory and for EDC when error correction is performed on a word of 64 bits (see chart

above). Unless EDC is implemented with 64-bit word, EDC is more costly than parity in number of DRAMs required to support it.

Performance is another attribute of the board that can suffer when EDC is implemented. The logic to perform EDC is understandably much more complicated; additional gates require more time for execution. However, with a 64-bit error correction scheme, the overhead of error checking is only incurred once every four 16-bit words or once every two 32-bit words, in sequential accesses. Since 64 bits are latched simultaneously into a much faster register than the memory chip, the register acts as a small cache. As a result, there is a performance enhancement which effectively offsets the increased time required for the additional logic.

Overall, EDC can translate into a significant improvement in error protection, and, as long as the EDC is implemented with a 64-bit word, the cost and performance tradeoff is insignificant.

Double-bit Error Detection and Correction: If One Is Good, Aren't Two Better?

Recently, some IC vendors have implemented logic in an EDC chip set that detects and corrects double-bit errors. This seems like a substantial improvement over single-bit correction and double-bit detection; in fact, the likelihood of two soft errors in one location is incredibly small and the benefit beset with operating negatives.

If an IC experiences a double-bit hard error, an EDC that automatically corrects a double-bit error is probably masking a bad IC that should be replaced. In addition, multiple-bit errors are never logged and the data integrity is jeopardized. For certain applications (satellites, for instance), where continuing operation is more important than complete data integrity, double-bit EDC will make a significant contribution. But, many system managers would prefer the assurance of data integrity.

EDC ON ARRAY CARDS

EDC Functionality

Most systems that implement EDC use separate memory controllers and arrays; the controller board implements the EDC while the arrays simply contain the extra storage to hold the check bits. Likewise, parity memory controllers have a CSR designed for parity. They only latch the address of data in the case of bad parity. Should an EDC memory card be placed in the array, the controller CSR only provides information on double- or multiple-bit errors that halt the system. To monitor the error correction on the memory array, a separate CSR is needed.

Currently, there are some separate controller/array systems that do not have EDC on the controller, but for which vendors sell EDC memory. This is now being advertised in the MicroVAX II marketplace. Since EDC works in a fundamentally different way than parity, customers should look very carefully at what they are buying. It is possible to decrease system reliability if the controller board does not provide adequate EDC functionality.

EDC Diagnostics

Another problem is diagnostics. If the system memory diagnostics are designed to test parity, they will yield little useful information on the functionality of an EDC array. Vendors of the EDC array offer a mechanical switch to toggle when diagnostics are being run, turning off the EDC. The result is that there is really no way to test the EDC array on the system; a custom tester is required to tell if the EDC is even working as advertised.

Worse, some boards only reveal single-bit errors using an LED, with no error logging or clearing through software. This requires opening the system box periodically to check for single-bit errors. If a soft error has occurred, the system must be powered down; the switch toggled; the cabinet closed; and the system powered up again before it is operational as an EDC memory.

Any realist knows that the probability of crashing a system increases by several orders of magnitude each time it is manually altered. An EDC system that requires a great deal of manual intervention is clearly more trouble than it is good. For those who live by the adage "If it ain't broke, don't fix it," make sure you know what you are buying.

REDUNDANCY

Redundancy is currently the ultimate error protection available for hardware. Many major vendors now offer systems in which all boards are "duplexed" — i.e., redundant—and the crash of either will not bring the system to a halt. Stratus Computer even has the system automatically call up remotely to order a replacement for a failing board. Customers only find out there was a failure when a replacement module shows up on their dock.

This is great, but curiously the systems still crash. Usually the reason is software: all the redundant hardware in the world cannot eliminate crashes due to bugs in system software. The operating system environments for redundant systems are necessarily more complex, resulting in a somewhat greater vulnerability to unpleasant encounters with wayward applications. Once an environment is stable, however, the added hardware reliability can reduce downtime to minutes a year.

Redundant systems are more than twice as expensive as simple EDC, because of the complexity of the operat-

ing environment. Still, with hardware rapidly declining in price relative to five years ago, the redundancy may be worth the cost.

Measurement of Reliability

Many people try to quantify reliability, and make meaningful comparisons based on these numbers. Because reliability involves statistical probabilities, talking in terms of individual boards or systems is meaningless. It is always important to remember that just because you can measure something, that does not mean that it is what you are looking for. The favorite joke on this runs:

A woman walking down the street one night comes across a man looking on the ground underneath a street light.

She asks him what he is up to, and he replies that he is looking for his lost car keys.

"Where did you lose them?" she asks.

"Over next to my car," he answers, pointing to his car up the street.

"So why are you looking over here?" the woman asks, incredulously.

"Because it's too dark to see anything over there."

Mean Time Between Failures

So goes the saga of measuring reliability. MTBFs are easy to compute, but splitting hairs over differences is really looking at the wrong issue. MTBFs are computed based upon the cumulative probabilities of failure due to wear-out of ICs and PC boards in a normal operating environment. When the probability of an individual component failing is very small (1 in a million years, or thereabouts), the probability of 1,000 or 5,000 such devices failing is approximated by the sum of the individual probabilities. To compute an MTBF, the manufacturer adds up the individual probabilities for all the ICs on the board, the PC board itself (based on the number of holes and layers), and each solder connection. This cumulative probability is the reciprocal of the MTBF.

The cumulative probabilities are very low, as one would expect. A memory vendor computes MTBFs from information provided by the suppliers of the components. Not that the suppliers have any conflict of interest, but they like to measure their component's reliability under circumstances that show it "in a good light." Usually, this means tightly controlled temperature and environment, complete static protection, and elimination of failures due to "abnormal" circumstances.

The good news is that the probability of failure in these circumstances is very low. The bad news is that "abnormal" circumstances are the cause of 99% of the failures. Poor static control in handling, improper soldering, conductive dust build-up, inadequate QC at the component level, poor quality printed circuit boards, corrosive desoldering fluids, incorrectly labelled ICs and dis-

cretes, out-of-revision programmable devices, improper packaging and shipping, and incorrect installation, to name a few, are all abnormal circumstances.

MEAN TIME TO REPAIR

Another index of reliability is the MTTR, since downtime is a function of both the MTTR and the MTBF. MTTR is simply the time it takes to repair a failure on a board. It is computed by estimating the probability of each failure mode and multiplying this by the expected time for repairing that failure. While this may be useful for the vendor to compute his labor costs in repair, it has very little to do with what a customer will experience.

Repair time depends primarily on whether a customer can perform on-site repair or whether the board must be returned to the factory. Most customers return all boards for repair, since this is really the best way of guaranteeing that it is done correctly. For customers willing to repair failed memory chips, the only issue is whether the memory is socketed or soldered-in. On-site repair of soldered-in memory is never a good idea.

MTTR then becomes dependent on vendor response time. If the vendor has implemented true 24-hour advance replacement, this becomes the MTTR.

REFERENCES AND SITE VISITS

In reality, there is no substitute for references and first-hand experience in determining vendor reliability. The vendor knows what his reliability problems are, and his larger customers will too. Vendors have been known to use a lot of poetic license in describing their return rates, so asking will not generally reveal the skeletons. Large customers, however, are usually willing to share their results. If a customer has a large sample size (200–500 boards) and a long history with the vendor you can usually assume they have seen some dirty laundry.

THE BOTTOM LINE

Reliability is an important factor in any system configuration. But the real bottom line is understanding what can or can't be done to ensure the maximum reliability for your investment. How much reliability is enough? It is probably a question that can never be answered entirely. However, understanding the risks is half the battle. With adequate knowledge, a reasonable decision can be made.

So goes the saga of measuring reliability. MTBFs are easy to compute, but splitting hairs over differences is really looking at the wrong issue. MTBFs are computed based upon the cumulative probability of failure due to wear-out of ICs and PC boards in a normal operating environment. When the probability of an individual component failing is very small (1 in a million years, or thereabouts), the probability of 1,000 or 5,000 such devices failing is approximated by the sum of the individual probabilities. To compute an MTBF, the manufacturer adds up the individual probabilities for all the ICs on the board, the PC board itself (based on the number of holes and layers), and each solder connection. This cumulative probability is the reciprocal of the MTBF.

The cumulative probabilities are very low, as one would expect. A memory vendor computes MTBFs from information provided by the suppliers of the components. Not just the suppliers have any control of interest, but they like to measure their component's reliability under circumstances that show it in a good light. Usually, this means tightly controlled temperature and environment, complete static protection, and elimination of failures due to abnormal circumstances.

The good news is that the probability of failure in these circumstances is very low. The bad news is that abnormal circumstances are the cause of 99% of the failures. Poor static control in handling, improper soldering, conductive dust buildup, inadequate I/O at the component level, poor quality printed circuit boards, corrosive desoldering fluids, incorrectly labelled ICs and dis-

Redundancy is currently the ultimate error protection available for hardware. Many major vendors now offer systems in which all boards are "duplicated"—i.e., redundant—and the crash of either will not bring the system to a halt. Stratus Computers even has the system automatically call up remotely to order a replacement for a failing board. Customers only find out there was a failure when a replacement module shows up on their dock.

This is great, but obviously the systems still crash. Usually the reason is software: all the redundant hardware in the world cannot eliminate crashes due to bugs in system software. The operating system environments for redundant systems are necessarily more complex, resulting in a somewhat greater vulnerability to unpleasant encounters with wayward applications. Once an environment is stable, however, the added hardware redundancy can reduce downtime to minutes a year.

Redundant systems are more than twice as expensive as simple EDC, because of the complexity of the oper-

ERROR DETECTION AND CORRECTION SYSTEM ARCHITECTURES

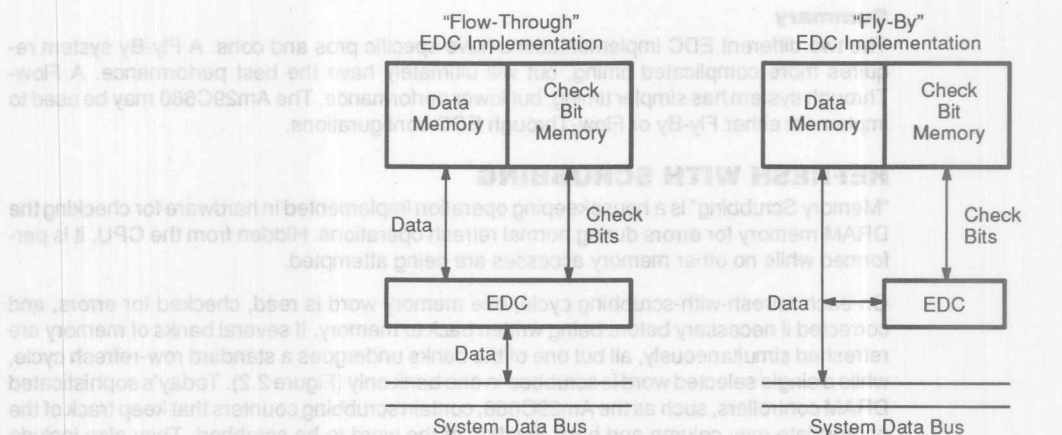
Once the decision to use EDC has been made, a designer must choose how this capability will be implemented within his memory system. In general, there are two distinct implementations of EDC, commonly referred to as *Fly-By* and *Flow-Through* (Figure 2.1).

FLY-BY

Fly-By, which is another term for a Check-Only Configuration, checks all words read from the memory to the bus for errors. If an error is discovered, it is flagged by the EDC and a recovery routine is initiated to complement the bit-in-error before the full data word is presented as valid. The erroneous word is gated off the bus and the corrected data substituted.

While this error-checking routine is being undertaken by the EDC circuit, the data words are simultaneously sent directly to the bus. The system always assumes that no error has occurred, a correct assumption in the majority of cases. This implementation provides maximum speed for most memory accesses; the cycle will occur without any delay from the EDC, since most words read from memory contain no errors.

However, if an error is detected, the CPU uses the Error or Multiple Error flags from the EDC to either interrupt the memory cycle or stretch it by adding wait states; this causes a significant throughput delay. At the designer's option, the corrected data may be written back to the memory if the EDC correction logic is enabled. In some instances, one may not wish to write the corrected data back to memory, such as in the case of a system employing memory "scrubbing" during refresh cycles (a more detailed explanation of memory scrubbing follows). In this case, the system automatically corrects the error later.



11580-005A

Figure 2.1 EDC Implementations

The Flow-Through method assumes each word is erroneous and completes a correction cycle during every memory access. This simplifies system timing because each memory cycle is identical. The Fly-By implementation only interrupts the memory cycle if an error is detected, and a recovery routine is initiated. This provides maximum speed for most memory accesses, since there is no EDC delay if an error has not occurred.

Commonly, a penalty of one wait state is the result of a correctable error. For non-correctable errors and instruction fetches, all high-performance processors have some form of pipelining or prefetch buffering. Since the corrupted instruction, at best, only gets to the decode stage of the processor before an interrupt is asserted, a recovery routine is executed before the error

is encountered. For data accesses, invalid data is read into the processor. However, after insertion of an interrupt, the processor will *not* crash due to invalid data.

Using the Fly-By implementation, data Reads proceed as fast with EDC as without. Slow-down occurs only if there is an error. Even if the memory system has an error every hour, this would only occur once every 3–4 *billion* memory cycles. So even with a high error rate, EDC in the Fly-By configuration has essentially zero impact on memory-system speed. Fly-By can be implemented with the 680XX and 80X86 processor families in addition to many other CPU and system-bus configurations.

FLOW-THROUGH

A Flow-Through EDC implementation places the EDC directly between the memory and system data bus. Using this configuration, also called “Correct Always,” all words from memory are assumed to contain errors and are routed first through an EDC circuit before reaching the bus. The EDC also assumes every word has an error and completes a correction cycle, which simplifies system timing. In this implementation, the impact of a memory error is negligible, for the EDC corrects the error without having to interrupt the system cycle. The EDC correction cycle runs at the same speed whether or not the data contains an error. All memory access cycles are slower due to the additional time required by the EDC. Usually, the Flow-Through configuration is used with microprocessors that have ample memory-timing budgets, or in systems that do not support bus-cycle restarts.

The increased speeds of today’s EDC products are continually making this EDC configuration more practical for high-end systems, since error detection and correction has become an insignificant part of the total memory-cycle time. The Am29C660E, for example, detects and corrects an error in a 32-bit data word in a mere 14 ns. This is a five times improvement in speed over circuits available several years ago.

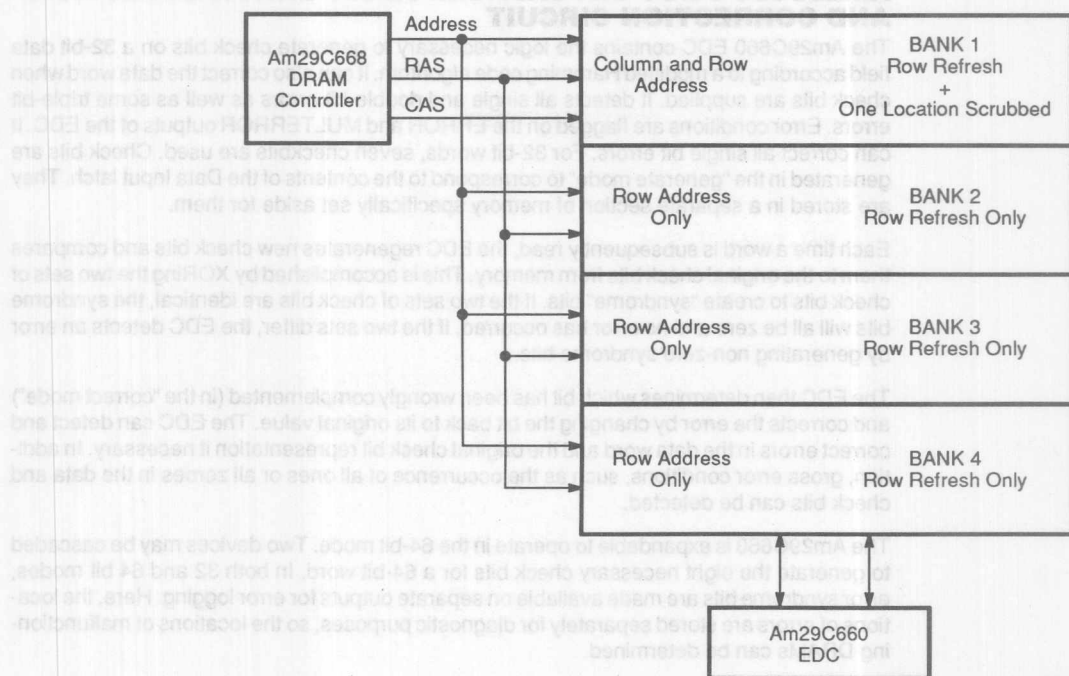
Summary

The two different EDC implementations have specific pros and cons. A Fly-By system requires more complicated timing, but will ultimately have the best performance. A Flow-Through system has simpler timing, but lower performance. The Am29C660 may be used to implement either Fly-By or Flow-Through EDC configurations.

REFRESH WITH SCRUBBING

“Memory Scrubbing” is a housekeeping operation implemented in hardware for checking the DRAM memory for errors during normal refresh operations. Hidden from the CPU, it is performed while no other memory accesses are being attempted.

On each refresh-with-scrubbing cycle, one memory word is read, checked for errors, and corrected if necessary before being written back to memory. If several banks of memory are refreshed simultaneously, all but one of the banks undergoes a standard row-refresh cycle, while a single selected word is scrubbed in one bank only (Figure 2.2). Today’s sophisticated DRAM controllers, such as the Am29C668, contain scrubbing counters that keep track of the appropriate row, column and bank location of the word to be scrubbed. They also include EDC initialization logic that writes a known value to all memory locations during power-up. For a 16-Mword memory (2^{24} locations) employing megabit DRAMs and one refresh every 16 μ s, scrubbing the entire memory takes four and one half minutes, regardless of the word width.



During scrubbing, one location of memory is checked for errors during normal refresh cycles. In this case, a single address location within bank #1 is scrubbed while the active row is refreshed in all four banks. The Am29C668 contains row, column, and bank counters that scrub all memory locations in succession via the Am29C660 EDC.

Figure 2.2 Memory Scrubbing

When an error occurs, a Read/Modify/Write R/M/W cycle is performed. The duration of a R/M/W cycle is longer than a normal Read or Write cycle. During refresh operations, a row in each bank is accessed by asserting the Row Address Strobe RAS line. This refreshes all locations in that row. If an error is detected, a Write operation is performed within the refresh cycle; wait states may be required to extend the cycle. However, system reliability is increased because soft errors cannot accumulate in areas of memory that are not frequently accessed.

When performing RAS-only refresh without scrubbing, all four RAS lines are activated, but the CAS lines remain inactive. A refresh with scrubbing cycle activates all four RAS lines and a single CAS line. Correctable errors detected during scrubbing cycles are not reported to the CPU.

AM29C660 CMOS CASCADABLE 32-BIT ERROR DETECTION AND CORRECTION CIRCUIT

The Am29C660 EDC contains the logic necessary to generate check bits on a 32-bit data field according to a modified Hamming code algorithm. It can also correct the data word when check bits are supplied. It detects all single and double-bit errors as well as some triple-bit errors. Error conditions are flagged on the ERROR and MULTERROR outputs of the EDC. It can correct all single bit errors. For 32-bit words, seven checkbits are used. Check bits are generated in the "generate mode" to correspond to the contents of the Data Input latch. They are stored in a separate section of memory specifically set aside for them.

Each time a word is subsequently read, the EDC regenerates new check bits and compares them to the original check bits from memory. This is accomplished by XORing the two sets of check bits to create "syndrome" bits. If the two sets of check bits are identical, the syndrome bits will all be zero and no error has occurred. If the two sets differ, the EDC detects an error by generating non-zero syndrome bits.

The EDC then determines which bit has been wrongly complemented (in the "correct mode") and corrects the error by changing the bit back to its original value. The EDC can detect and correct errors in the data word and the original check bit representation if necessary. In addition, gross error conditions, such as the occurrence of all ones or all zeroes in the data and check bits can be detected.

The Am29C660 is expandable to operate in the 64-bit mode. Two devices may be cascaded to generate the eight necessary check bits for a 64-bit word. In both 32 and 64 bit modes, error syndrome bits are made available on separate outputs for error logging. Here, the locations of errors are stored separately for diagnostic purposes, so the locations of malfunctioning DRAMs can be determined.

The Am29C660 also includes two diagnostic modes in which diagnostic data may be forced into portions of the chip via software to simplify device testing and to execute system diagnostic functions. Using this feature, "dummy" data words and check-bit representations may be loaded into their respective input latches to generate different error detection and correction operations.

PROGRAM TO EVALUATE AM29C660 MULTIPLE ERROR-DETECTION CAPABILITY

A program was written to evaluate the error-detection capability of the Am29C660. It simulates all the possible combinations of bit errors. The number of possible errors for a given number of bits in error is a combinatorial function of the number of bits in the word and the number of bits in error. If m is the number of bits in error and n is the total number of bits in the code word, then the total number of words is calculated as follows:

$$\frac{n!}{m!(n-m)!}$$

where $n! = 1 \times 2 \times 3 \times \dots \times n$.

The program generates the data and check bits with the specified number of bits in error. The program calculates the check bits for the data and the calculated check bits are XOR-ed with the check bits generated in the previous step. If the resulting syndrome bits are zeros, the errors were not detected. If the syndrome bits are not zeros, they must be compared against the syndromes for a single-bit error through a look-up table. If the syndrome bits match a single-bit error, the word would be "miscorrected." The EDC thinks the error is a single-bit error and attempts to correct it. If none of the previous conditions are satisfied, the error is detected. This process is performed for all the possible combinations.

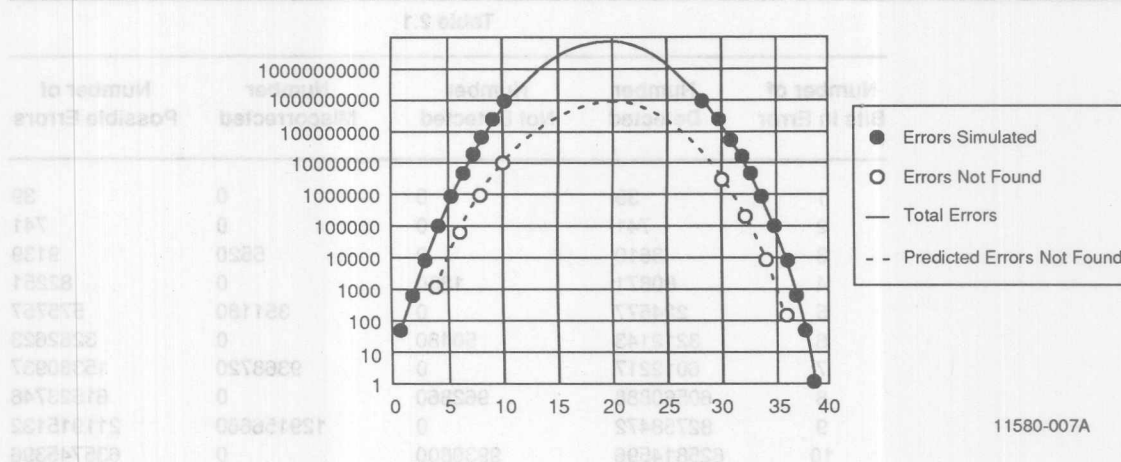
Table 2.1 summarizes the results of the simulation. The time needed to execute the program limited the number of bits in error that could be simulated. From the table, it can be seen that for any odd number of errors, all the errors are detected; but some are miscorrected. For an even number of bits in error, none are miscorrected; but some are not detected.

Table 2.1

	Number of Bits in Error	Number Detected	Number Not Detected	Number Miscorrected	Number of Possible Errors
	1	39	0	0	39
	2	741	0	0	741
	3	3619	0	5520	9139
	4	80871	1380	0	82251
	5	224577	0	351180	575757
	6	3212143	50480	0	3262623
	7	6012217	0	9368720	15380937
	8	60560888	962860	0	61523748
	9	82758472	0	129156660	211915132
	10	625814596	9930800	0	635745396
	29	248334700	0	387410696	635745396
	30	208604844	3310288	0	211915132
	31	24031156	0	37492592	61523748
	32	15140312	240625	0	15380937
	33	1275240	0	1987383	3262623
	34	566845	8912	0	575757
	35	32075	0	50176	82251
	36	8983	156	0	9139
	37	273	0	468	74
	38	39	0	0	39
	39	1	0	0	1

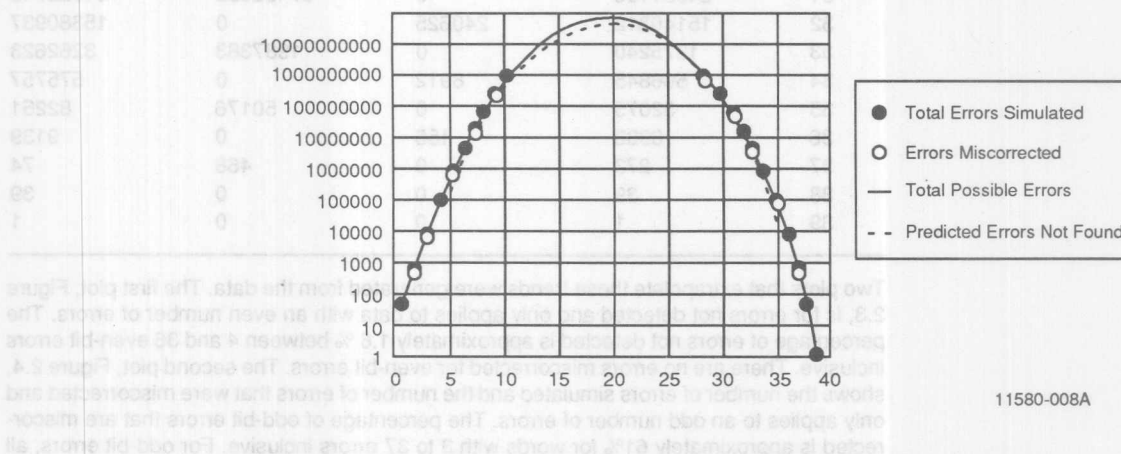
Two plots that extrapolate these trends were generated from the data. The first plot, Figure 2.3, is for errors not detected and only applies to data with an even number of errors. The percentage of errors not detected is approximately 1.6 % between 4 and 36 even-bit errors inclusive. There are no errors miscorrected for even-bit errors. The second plot, Figure 2.4, shows the number of errors simulated and the number of errors that were miscorrected and only applies to an odd number of errors. The percentage of odd-bit errors that are miscorrected is approximately 61% for words with 3 to 37 errors inclusive. For odd-bit errors, all errors are detected.

For nibble memories, assuming a single failure of one DRAM, the probability of detecting the error is 92.5%; the probability of not detecting an error is 1.5%; and the probability of miscorrecting an error is 6.0%. This assumes that the number of 1-, 2-, 3- and 4-bit errors are equally distributed, which should be the case for a total failure.



11580-007A

Figure 2.3. Number of Errors Not Detected for Even Bits in Error



11580-008A

Figure 2.4. Number of Errors Miscorrected for Odd Number of Bits in Error

For multiple memories, assuming a single failure of one DRAM, the probability of detecting the error is 92.5%, the probability of not detecting an error is 7.5%, and the probability of miscorrecting an error is 0.0%. This assumes that the number of 1, 2, 3- and 4-bit errors are equally distributed, which would be the case for a total failure.

MEMORY RELIABILITIES WITH AND WITHOUT THE AM29C660 EDC CIRCUIT

DRAM manufacturers express MTBF in terms of failures in time (FITs). One FIT represents one error in one billion (10^9) hours of operation. Toshiba, a major supplier of 1-Mbit DRAMs, claims a soft-error rate of 252 FITs (one failure every 3.97×10^6 hours). Clearpoint, a manufacturer of add-in memory boards for a variety of systems and buses, estimates the soft-error rate at approximately 1000 FITs (one failure every 10^6 hours). The actual value is somewhere between these two figures. An analysis using both FIT rates follows.

The MTBF in hours for each separate DRAM is $\frac{10^9}{\text{FITs}}$

The memory system MTBF in hours is $\frac{\text{DRAM MTBF}}{\# \text{ of DRAMs}}$

The following table summarizes the MTBF for different memory sizes without EDC assuming a FIT rate of 252.

# of Memory Chips	Memory Size (Mbyte)	Memory MTBF (yrs)
32	4	14.1
64	8	7.1
96	12	4.7
128	16	3.5
160	20	2.8
192	24	2.4

The following table assumes a FIT rate of 1000 and no EDC.

# of Memory Chips	Memory Size (Mbyte)	Memory MTBF
32	4	3.6 yrs
64	8	1.8 yrs
96	12	1.2 yrs
128	16	326 days
160	20	260 days
192	24	217 days

With EDC, all single-bit errors are detected and corrected. The probability of a fatal two-bit error occurring depends upon several system considerations. The Am29C668 4-Mbit Configurable DRAM controller can be used to "scrub" the memory during refresh cycles. This insures the data integrity of seldom-accessed memory locations and increases the MTBF. If scrubbing is not used, the MTBF for the system depends upon when two soft errors occur in the same word. The approximation for the occurrence of two soft errors in the same word is given by the approximation to the birthday paradox*:

$$(\text{MTBF using 39 chips without EDC}) \times \sqrt{\frac{\pi \cdot \text{Number of Memory words}}{2}}$$

* The birthday paradox is that, on the average, only 24 people need be asked their birthday before two people are found to have been born on the same day of the year. This is a well known problem in statistics.

The Am29C660 adds an overhead of seven bits for every 32-bit word. For a 32-bit EDC system without scrubbing, the system MTBF, assuming a FIT rate of 252, is:

# of Memory Chips	Memory Size (Mbyte)	Memory MTBF (yrs)
39	4	14,907
78	8	10,541
117	12	8,607
156	16	7,454
195	20	6,667
234	24	6,086

The following table assumes a FIT rate of 1000 and EDC and no scrubbing.

# of Memory Chips	Memory Size (Mbyte)	Memory MTBF (yrs)
39	4	3,757
78	8	2,656
117	12	2,168
156	16	1,878
195	20	1,680
234	24	1,534

These numbers, however, are overly optimistic because these tables disregard any hard failures. The Am29C660 and the extra memory chips also impact the reliability of the system, because there are more memories subject to failure. The actual increase in MTBF is more on the order of 50 or 60, which increases the memory reliability in the worst case from 217 days to 30 years. If scrubbing is used, the MTBF is increased even more.

An EDC requires slightly greater overhead than a parity system, seven check bits versus four parity bits, but the EDC offers a dramatic increase in reliability. Parity only detects errors. EDC, therefore, is a very valuable tool in increasing system reliability.

With EDC, all single-bit errors are detected and corrected. The probability of a total two-bit error occurring depends upon several system considerations. The Am29C660 4-Mbit Controllable DRAM controller can be used to "scrub" the memory during refresh cycles. This ensures the data integrity of seldom-accessed memory locations and increases the MTBF. If scrubbing is not used, the MTBF for the system depends upon when two soft errors occur in the same word. The approximation for the occurrence of two soft errors in the same word is given by the approximation to the birthday paradox:

$$\text{MTBF using 39 chips without EDC} \propto \sqrt{\frac{\pi \times \text{Number of Memory Words}}{2}}$$

* The birthday paradox is first on the average only 24 people need be asked their birthday before two people are found to have been born on the same day of the year. This is a well-known problem in statistics.

SYSTEM BUSES

The section of this chapter, "Understanding Memory Design," discusses system-bus efficiency, particularly as it relates to memory, referring to examples such as Q-Bus and Multibus II. It is important that the designer be familiar with the various available system buses, since he may need to design interface circuitry to meet a strict set of specifications. There are many choices: some buses are designed for specific systems from DEC, IBM etc.; others are vendor independent and offer open standards. Choosing the right bus is rarely easy. The designer must consider many factors: e.g., board size, connector type, arbitration methods, protocols, available semiconductor technologies. A brief overview of the most popular buses, by no means a complete listing, is given here.

The VMEbus

The VMEbus was developed by Motorola, in association with other companies, to provide an open architecture. Perhaps the most popular bus among OEMs, the VME offers 8-, 16-, or 32-bit data and 16-, 24-, or 32-bit addressing and a 40 Mbyte/s bandwidth. The VMEbus is rapidly becoming the choice for military applications.

Numerous products are offered for use with the VMEbus, including almost all processors, memories and memory boards, controllers, error-detection and connection circuits, and other support products. The asynchronous VMEbus provides for blockmode and unaligned transfers; it is extremely flexible with minimal compatibility problems. Present plans include a 256-bit data path, 2-3 Gbyte/s bandwidth, 64-bit addressing and scalability.

VERSAbus was VME's predecessor, designed primarily for 68000-based systems. It is in limited use today.

Multibus I and II

Multibus I is one of the most popular single-board computer buses and boasts an extremely large installed base in both the military and OEM markets. Developed by Intel, it offers a simple architecture: 16-bit data, 24-bit addressing, asynchronous operation and requires no multiplexing. It is still an excellent choice for 8- and 16-bit applications; however the growth path stops here. Intel was forced to meet the versatility demands of more sophisticated systems; the result is Multibus II.

Multibus II is a synchronous bus with five levels of embedded sub-buses. The main parallel-system bus is 32 bits wide and operates at 40 Mbyte/s; the local memory bus is even faster. Combined with a serial bus, Multibus II architecture offers a broad range of bandwidths in one specification. Synchronous buses are usually tightly specified to keep compatibility problems to a minimum and to make system design easier. However, the five levels of sub-buses, all with different clock speeds, complicate the design. To solve this problem, Intel offers a bus-interface chip set to standardize bus communication.

Multibus II offers a broad capability for tightly coupled, synchronous operation of many processors and shared devices to avoid the problems of centralized arbitration.

The NuBus

The NuBus, another bus that supports 8-, 16- and 32-bit addressing, is simple, flexible and easy to use. It is best known for its application in the Macintosh II. NuBus has only one address space, as compared to three required by both Multibus II and VMEbus; only four control lines are required to define a transaction, compared to many more for Multibus and VMEbus. NuBus supports a 37.5 Mbyte/s data-transfer rate, as well as several types of DMA transactions.

A NuBus overhaul in 1990 may provide twice the current performance for 32-bit systems with a transfer rate to 80 Mbyte/s and improved specifications that will enhance its position in both the workstation and industrial markets.

AT Bus

The IBM PC, originally based on the 8088, is probably the most popular system ever built. There have been two major enhancements, the XT and the AT. The latter is based on the 80286 and offers a 16-bit data bus vs only an 8-bit bus on the XT and the PC. The AT architecture is adequate for single-user machines and continues to be a strong contender in the work-station marketplace. However, the AT bus is still only 16 bits wide, but the machine performance continues to improve with faster processors and peripherals. Most work-station vendors now offer some degree of compatibility through networking to provide for PC applications on larger machines.

Micro Channel

Micro Channel currently provides a low-end standard to improve upon the performance of the AT bus. It resides only in the PS/2 Model 70 and 80 systems. (See detailed discussion in the Application Note, "IBM PS/2 12-Mbyte Memory Board with Error Detection and Correction", Chapter 4). Micro Channel is an IBM-proprietary bus.

The Micro Channel architecture uses combinations of 8-, 16- and 32-bit connectors to implement a complex bus arbitration scheme for sharing address, data, and control lines without conflict. Bandwidth is 20 Mbyte/s.

EISA (Extended Industry Standard Architecture)

Based on the AT bus, EISA was designed by a consortium of computer vendors and is intended to be the answer to Micro Channel. It targets single-CPU environments and supports multiple-bus masters. EISA serves as a shared resource in a network of PCs and segments its architecture into memory and I/O buses. Bandwidth is 33 Mbyte/s.

EISA is a 32-bit bus which allows 16-bit add on cards, designed for AT Bus to be used, as well as new 32-bit EISA-only cards.

The Q-Bus

For many years, DEC's Q-Bus has been the most popular OEM bus for low-end applications, such as process control and single-user systems. Originally designed for the LSI-11 microcomputers, it is still used extensively throughout the MicroVAX computer family.

The asynchronous Q-Bus is inexpensive and simple to use. To save signal lines, address and data are multiplexed. The original 16 address lines have been increased to 22 bits to provide a 4-Mbyte address space. The data bus remains only 16 bits wide.

The Q-Bus specifies that DEC (or equivalent) line drivers be used on any bus interface. However, some designers have found ways to hang logic directly on the bus with no ill effects. Some years ago, DEC added block-mode DMA to speed data transfer. An on-board memory controller enables add-in vendors to offer a wide variety of memory types that can easily be used with the Q-Bus.

MicroVAX II

Another successful DEC bus is the MicroVAX II, which uses the Q-Bus for I/O but has a separate memory bus to reduce bus traffic and speed memory access. DMA is handled over the Q-Bus to the memory controller, which is on the processor card. The data is then transferred to the memory via the memory bus. Unlike the early MicroVAX I, the memory bus has a full 32-bit data path.

The memory bus is synchronous with a fixed 400-ns cycle time, which means that all memory boards perform the same; faster memories buy nothing. Parity is the standard level of error protection, providing error detection only, no correction. To exceed the 16-Mbyte maximum configuration of the MicroVAX II, some vendors offer RAM-disk expansion on the Q-Bus.

MicroVAX 3000

The MicroVAX 3000 Series adds many improvements: speed, error correction and memory addressability. DEC continues to expand its bus architectures to support the VAX product line.

Futurebus+

The IEEE specification for Futurebus+, the bus for 64- and 128-bit, and even 256-bit, systems, was due for completion in the fourth quarter '89. Using backplane transceiver logic (BTL), designed specifically for driving a backplane bus, Futurebus+ solves the transmission line problems that limit the speed for buses based on TTL. It boasts speeds of 400 to 3200 Mbyte/s, depending on the size of the system, 32 to 256 bits. Futurebus+ uses an asynchronous-bus-interface protocol that is completely independent of the processor family or technology used to implement the system. The bus contains extensive monitoring, diagnostic and error-detection facilities.

Futurebus+ also features a distributed arbitration scheme that expedites the building of fault tolerance into a system. It is an open-architecture bus that also includes a broadcast/broadcast facility for interacting back and forth with multiple boards. The bus supports the implementation of a variety of caching methodologies within a single shared-memory multiprocessing system.

There will probably be very few Futurebus+-only systems appearing in the marketplace for some time. Most people will experiment by using Futurebus+, to upgrade VME or Multibus II systems. Bridges or links will give users a variety of I/O, peripheral and lowcost CPU cards for tasks that do not require the full power of Futurebus+. Many companies are exploring ways to bridge the gap between their buses and Futurebus.

Opinions vary on the role the VMEbus will play in the future. Most board vendors, for the time being, will probably provide bridges as stepping stones to link VME with Futurebus+. Intel and the Multibus Manufacturer's Group plan to link Multibus II to Futurebus+.

Users must be able to upgrade their installed systems by connecting them via cable to the new high-performance system, thus preserving their investments in both hardware and software.

REFERENCES

Electronic Engineering Times, CMP Publications, August 14 and September 11, 1989.

The Designer's Guide to Add-In Memory, Third Edition 1989, Chapter Five, *The Final Step—An Industry Survey*, Clearpoint Research Corporation, Hopkinton, MA.



CHAPTER 3

Microprocessor Interfaces to the Am29C668 Configurable Dynamic Memory Controller/Driver

Introduction	3-3
Am29C668 CDMC to Am29000 Streamlined Instruction Processor Interface	3-5
Am29C668 CDMC to 80C286 Microprocessor Interface	3-41
Am29C668 CDMC to Am386™ Microprocessor Interface	3-57
Am29C668 CDMC to 68020 Microprocessor Interface	3-73
Am29C668 CDMC to 80486 Microprocessor Interface	3-93

CHAPTER 3

Microprocessor Interfaces to the Am29C668 Configurable Dynamic Memory Controller/Driver

3-2	Introduction
3-3	Am29C668 CDMC to Am8000 Streamlined Instruction Processor Interface
3-4	Am29C668 CDMC to 80C286 Microprocessor Interface
3-5	Am29C668 CDMC to Am86™ Microprocessor Interface
3-6	Am29C668 CDMC to 80859 Microprocessor Interface
3-7	Am29C668 CDMC to 80486 Microprocessor Interface



Microprocessor Interfaces to the Am29C668 Configurable Dynamic Memory Controller/Driver

INTRODUCTION

This chapter includes application notes describing five different interface designs utilizing the Am29C668 4M Configurable Dynamic Memory Controller/Driver (CDMC) and five well-know microprocessors—the Am29000, the 68020, the 80C286, the 80486, and the Am386™ microprocessor. The CDMC acts as the address controller between the microprocessor and the dynamic memory array, providing control for 4M, 1M, 256K and 64K dynamic RAMs.

Each interface was designed to provide maximum performance at reasonable cost. Each is as general as possible so that the user may tailor his implementation to a specific memory system. Possible changes are discussed with associated system requirements and implications. A block diagram, timing analysis, and logic equations necessary to implement each design are included.

Several of these application notes demonstrate the high-throughput memory performance obtainable from cost effective DRAM-only memory designs without requiring expensive SRAM caches for instructions and/or data. Memory accessing features such as page mode accessing and bank interleaving, which are available on the Am29C668 CDMC, significantly enhances DRAM system performance.

Microprocessor Interfaces to the Am29C668 Configurable Dynamic Memory Controller/Driver

INTRODUCTION

This chapter includes application notes describing five different interface designs utilizing the Am29C668 4M Configurable Dynamic Memory Controller/Driver (CDMC) and five well-known microprocessors—the Am29000, the 68020, the 80C86, the 80486, and the Am386™ microprocessor. The CDMC acts as the address controller between the microprocessor and the dynamic memory array, providing control for 4M, 1M, 256K and 64K dynamic RAMs.

Each interface was designed to provide maximum performance at reasonable cost. Each is as general as possible so that the user may tailor the implementation to a specific memory system. Possible changes are discussed with associated system requirements and implications. A block diagram, timing analysis, and logic equations necessary to implement each design are included.

Several of these application notes demonstrate the high-throughput memory performance obtainable from cost effective DRAM-only memory designs without requiring expensive SRAM caches for instructions and/or data. Memory accessing features such as page mode accessing and bank interleaving, which are available on the Am29C668 CDMC, significantly enhance DRAM system performance.

Am29C668 Configurable Dynamic Memory Controller to Am29000 Streamlined Instruction Processor Interface

INTRODUCTION

The interface between the Am29C668 4-Mbit Configurable Dynamic Memory Controller (CDMC) and the Am29000 Streamlined Instruction Processor was designed for maximum performance, while using relatively inexpensive DRAMs. This design uses 100 ns fast-page-mode DRAMs, yet achieves single-cycle burst accesses at 20 MHz. It also uses a minimum number of devices to reduce the required board space. This design is as general as possible so that users may tailor their implementations to specific memory systems. A block diagram, timing analyses and logic equations necessary to implement the design are included.

DISTINCTIVE CHARACTERISTICS

- Am29C668 4-Mbit Configurable Dynamic Memory Controller/Driver with Auto Timing
- 20-MHz Am29000 Streamlined Instruction Processor
- 100-ns Fast-Page-Mode 1 Mbit x 1 DRAMs. Also supports 256 Kbit x 4 Fast-Page-Mode DRAMs or 256 Kbit x 1 Fast-Page-Mode DRAMs

- Single Am29C668 Controls Two Banks of Interleaved Memory
- 8-Mbyte Dynamic Memory per Am29C668
- Four-Cycle Initial Access on Read Cycles, Three-Cycle Initial Access on Write Cycle, Single-Cycle Burst Accesses for Read and Write Cycles. Three-Cycle Read Access within a Page, Two-Cycle Initial Write Access within a Page
- Supports Instruction Burst Restart, Two-Cycle Access
- Supports Byte Writes
- Supports 4 Gbyte Address Space each for Instruction Memory, Data Memory and I/O. Separate Instruction and Data Memory, with Instruction Memory Accessible via the Data Bus to Load Programs
- Compatible with the Adapt29K™ In-Circuit Diagnostic System by Decoding Option Bits (OPT[2:0])

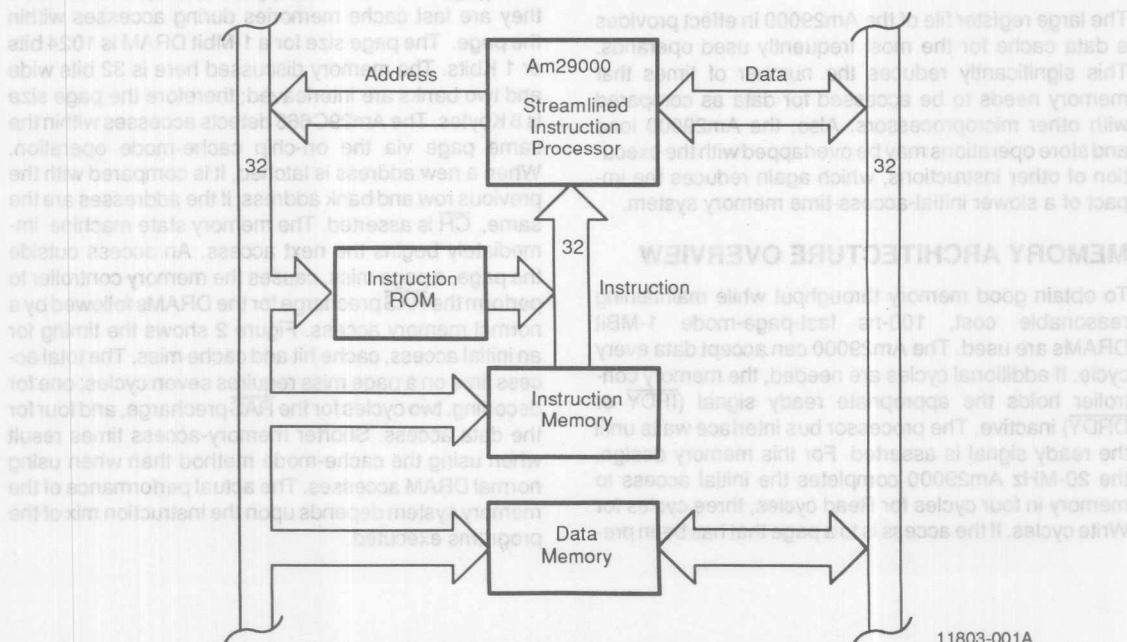


Figure 1. Am29000 System Diagram

Publication #	Rev.	Amendment	Issue Date
11803	B	/0	8/91

Am29000 OVERVIEW

When compared with static RAM (SRAM), dynamic RAM can provide far more memory at lower cost and power in the available board space. The main penalty in using DRAM is a loss of speed in the initial memory-access time. Burst-access performance can be maintained by using bank interleaving and fast-page-mode DRAMs. Fortunately, the Am29000 provides features that help compensate for a slower initial access time of system memory.

The Am29000 has an external Harvard architecture with separate data and instruction buses (Figure 1) so that the processor can fetch instructions and data simultaneously. With slower memories, it becomes important to maintain separate instruction and data spaces to increase the probability of instruction and data accesses occurring simultaneously, while decreasing the probability of a data access preempting an instruction burst. The Am29000 also has burst-mode loads, stores and instruction accesses to provide maximum memory bandwidth.

The Am29000 branch target cache (BTC) stores the first four instructions after a successful branch. The BTC is two-way, set associative, with 16 blocks per set and a block size of four words; there are 512 bytes of storage or 128 words. When a branch is taken, the first four instructions come from the BTC if the branch target address is in the cache. At the same time, the first instruction following those in the cache are accessed. The first three cycles of the initial memory access are hidden by the execution of the instructions in the BTC.

The large register file of the Am29000 in effect provides a data cache for the most frequently used operands. This significantly reduces the number of times that memory needs to be accessed for data as compared with other microprocessors. Also, the Am29000 load and store operations may be overlapped with the execution of other instructions, which again reduces the impact of a slower initial-access-time memory system.

MEMORY ARCHITECTURE OVERVIEW

To obtain good memory throughput while maintaining reasonable cost, 100-ns fast-page-mode 1-Mbit DRAMs are used. The Am29000 can accept data every cycle. If additional cycles are needed, the memory controller holds the appropriate ready signal ($\overline{\text{IRDY}}$ or $\overline{\text{DRDY}}$) inactive. The processor bus interface waits until the ready signal is asserted. For this memory design, the 20-MHz Am29000 completes the initial access to memory in four cycles for Read cycles, three cycles for Write cycles. If the access is to a page that has been pre-

viously accessed, a Read access is completed in three cycles and a Write in two cycles. During Write accesses, the ready signal is asserted one cycle earlier than in the Read cycle because the data is latched from the bus. The memory cycle still requires the same number of cycles to complete the actual memory access, but the system bus is freed to start another access.

There are four major types of speciality-mode DRAMs: fast-page-mode, static-column, nibble-mode and video DRAMs (VRAMs). Nibble-mode DRAMs can access four bits of data in a modulo-4 fashion, but are not applicable to this design because the length of the burst is indeterminate. VRAMs are attractive, since they have an on-chip shift register that permits concurrent data and instruction accesses; however they have cost and availability disadvantages. Static-column DRAMs (SCDRAMs) have a simpler interface since only the column address is changed to access another location in memory. However, this advantage presents a drawback. Since a fast-page-mode DRAM latches the row and column address, the address may change much sooner than an address in an SCDRAM, for which the column address must remain stable until the data is latched externally. This means that the page-mode DRAM can effectively overlap the address propagation delay with the memory access time, thereby giving better performance for comparable-speed DRAMs. In addition, to obtain the 20 MHz throughput using 100-ns SCDRAMs, each bank must be controlled by one Am29C668, thereby increasing control logic, cost and required board space.

Fast-page-mode DRAMs appear to the processor as if they are fast cache memories during accesses within the page. The page size for a 1-Mbit DRAM is 1024 bits or 1 Kbits. The memory discussed here is 32 bits wide and two banks are interleaved; therefore the page size is 8 Kbytes. The Am29C668 detects accesses within the same page via the on-chip cache-mode operation. When a new address is latched, it is compared with the previous row and bank address; if the addresses are the same, $\overline{\text{CH}}$ is asserted. The memory state machine immediately begins the next access. An access outside the page, a page miss, causes the memory controller to perform the $\overline{\text{RAS}}$ precharge for the DRAMs followed by a normal memory access. Figure 2 shows the timing for an initial access, cache hit and cache miss. The total access time on a page miss requires seven cycles: one for decoding, two cycles for the $\overline{\text{RAS}}$ precharge, and four for the data access. Shorter memory-access times result when using the cache-mode method than when using normal DRAM accesses. The actual performance of the memory system depends upon the instruction mix of the programs executed.

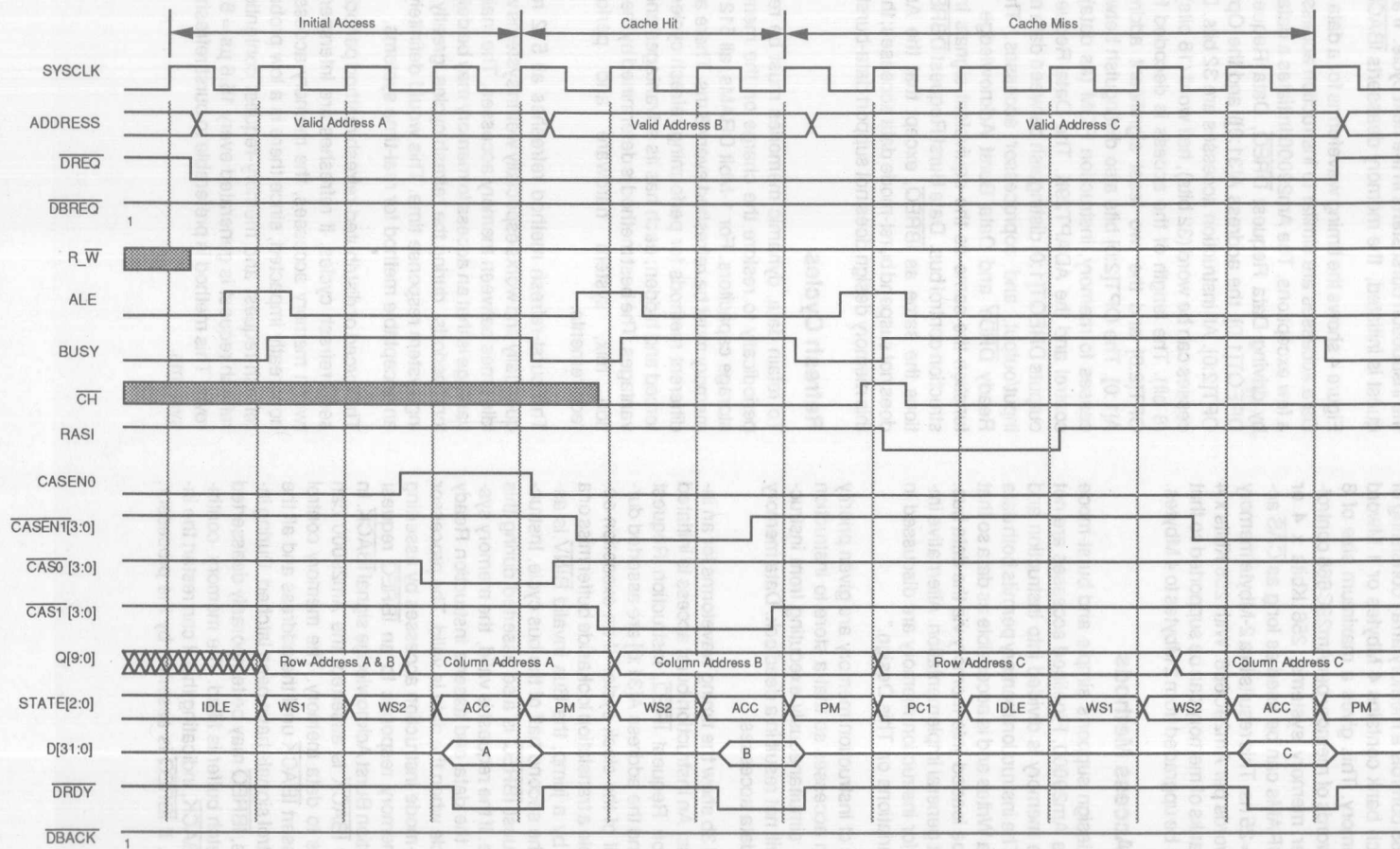


Figure 2. Cache-Mode Timings

11803-002A

Each Am29C668 controls a memory array consisting of two banks. Each bank contains 4 Mbytes or 1 Mword (32-bits) of memory. This gives a maximum size of 8 Mbytes or 2 Mwords of memory per Am29C668 controller. For smaller memory systems, 256 Kbits x 4 or 256 Kbits x 1 DRAMs can be used as long as $\overline{\text{CAS}}$ access time $t_{\text{CAC}} = 25$ ns. This results in a 2-Mbyte memory size or 512 Kwords per Am29C668. With 256 Kbits x 4 DRAMs, four banks of memory can be supported so that the system can be upgraded from 2 Mbytes to 4 Mbytes.

Supported Access Methods

This memory design supports simple and burst-mode accesses of the Am29000. Pipelined accesses are not supported. The memory is divided into instruction and data memory. The instruction memory permits both data Reads and data Writes and is accessible as data so that programs can be loaded into memory via the data bus. This is the most general implementation. Alternative implementations for instruction memory are discussed in the section "Variations on This Design."

Data accesses of instruction memory are given priority over instruction accesses so that a store to instruction memory, while simultaneously executing from instruction memory, will not result in a deadlock. Data memory supports only data accesses.

Figures 3a and 3b show the timing waveforms for an instruction access. An instruction burst access is initiated when Instruction Request $\overline{\text{IREQ}}$, Instruction Request Type $\overline{\text{IREQT}}$, and the address $A[31:0]$ are asserted during the first half of the clock cycle. If an exception occurs, for example a translation lookaside buffer miss or a jump followed by a jump, the Bus Invalid $\overline{\text{BINV}}$ is asserted during the second half of the bus cycle. Instruction Burst Request $\overline{\text{IBREQ}}$ is also asserted during this part of the cycle. If the request is valid, the memory system accesses the data and asserts Instruction Ready $\overline{\text{IRDY}}$ in the cycle when the data is valid. The processor requests burst-mode instruction accesses by asserting $\overline{\text{IBREQ}}$. The memory responds to an $\overline{\text{IBREQ}}$ request with the Instruction Burst Acknowledge signal $\overline{\text{IBACK}}$. In the cycle after $\overline{\text{IBACK}}$ is asserted, the Am29000 can start an access to data memory. The memory control logic cannot assert $\overline{\text{IBACK}}$ until the address and all the necessary control signals have been latched. During instruction bursts, $\overline{\text{IBREQ}}$ may be temporarily deasserted when the prefetch buffer is filled. The memory continues to assert $\overline{\text{IBACK}}$, indicating that it can restart the instruction burst. If $\overline{\text{IBREQ}}$ is asserted by the processor,

the instruction burst starts in the next cycle. If a new request is initiated, the memory deasserts $\overline{\text{IBACK}}$.

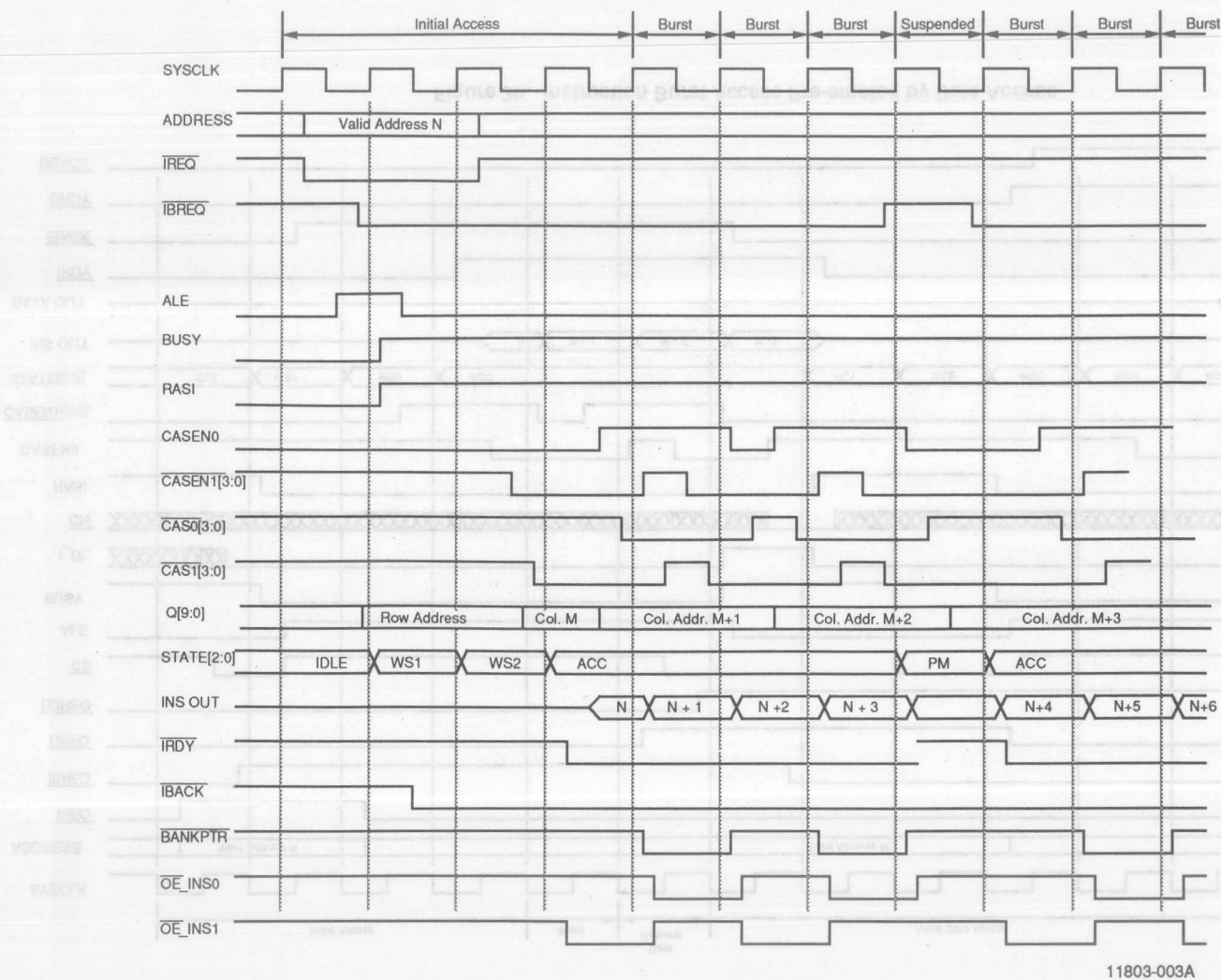
Figure 4 shows the timing waveforms for a data access. Data accesses are similar to instruction accesses, with a few exceptions. The Am29000 initiates a data access by driving Data Request $\overline{\text{DREQ}}$, Data Request Type $\overline{\text{DREQT}}[1:0]$, the address $A[31:0]$ and the Option bits $\text{OPT}[2:0]$. All instruction accesses are 32 bits. Data accesses can be word (32 bits), half word (16 bits) or byte (8 bit). The length of the access is decoded from the $\text{OPT}[2:0]$ and the two least significant address bits $A[1:0]$. The $\text{OPT}[2:0]$ bits also distinguish between accesses to memory, instruction ROM (as data), cache control and the ADAPT29K. The Data Request Type outputs $\overline{\text{DREQT}}[1:0]$ distinguish between data memory, input/output, and coprocessor accesses. The Data Ready $\overline{\text{DRDY}}$ and Data Burst Acknowledge $\overline{\text{DBACK}}$ function the same as the equivalent signals in the instruction control bus. Data Burst Request $\overline{\text{DBREQ}}$ functions the same as $\overline{\text{IBREQ}}$, except that the Am29000 does not suspend burst-mode data accesses; therefore, this memory design does not support data-burst restart.

Refresh Cycles

To retain data, dynamic memories must be refreshed periodically to restore the charge on the memory-cell storage capacitors. For 1-Mbit DRAMs, all 512 rows of memory must be refreshed every 8 ms. There are three different methods for performing refresh cycles: burst, forced and hidden; each has its advantages and disadvantages. The best method is determined by the instruction mix, system hardware and performance requirements.

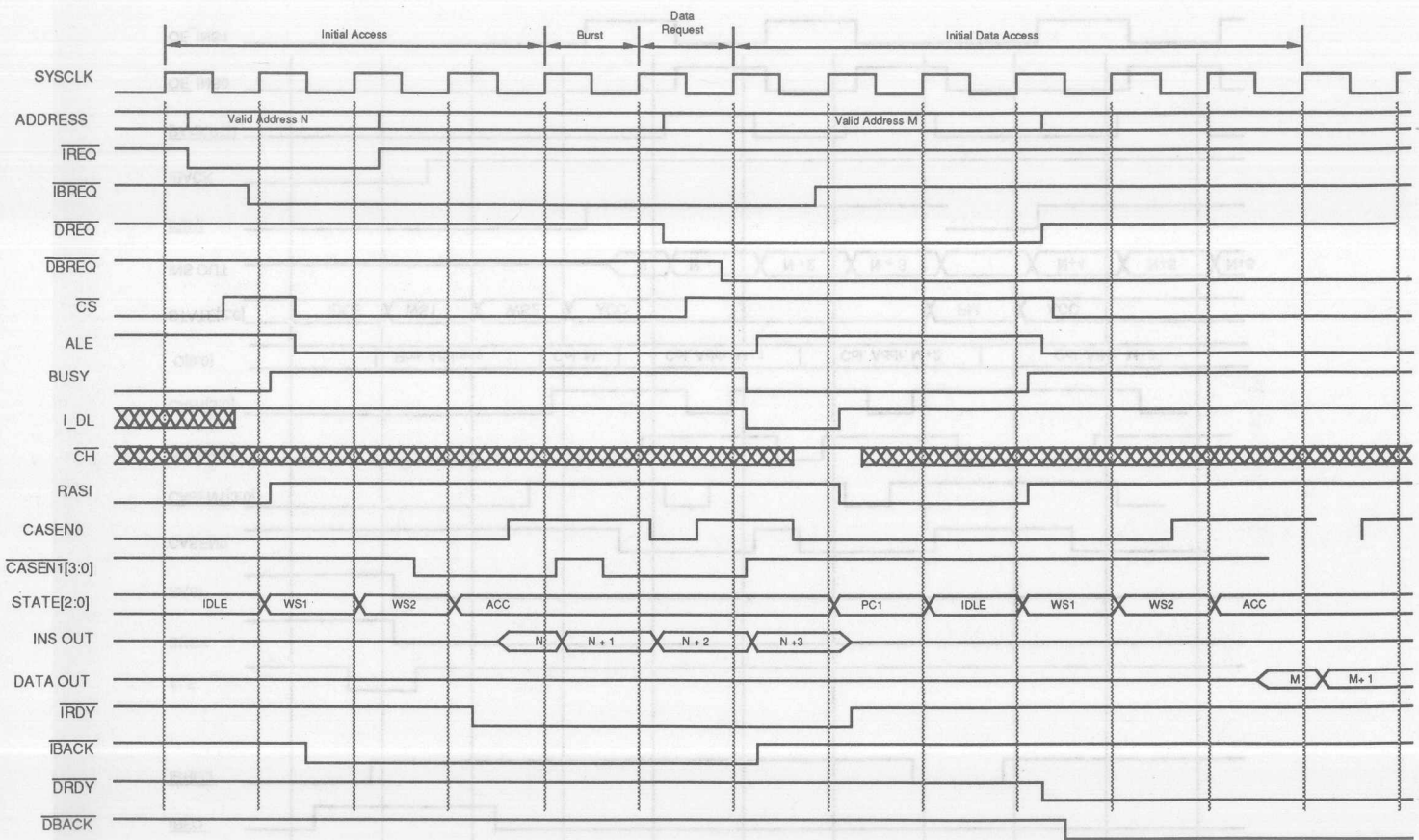
The burst-refresh method refreshes all 512 rows sequentially and works especially well in systems with long idle times between memory accesses. The main disadvantage is that an access to memory may be delayed for long periods during the refresh cycles, greatly impacting system response time. This would definitely not be an acceptable method for real-time systems.

The forced or distributed-refresh method periodically inserts refresh cycles. If refreshes are interspersed between memory accesses, the memory-access time is not greatly impacted, since there is a low probability of refresh-request and memory-request contention. One refresh request is generated every $15.6 \mu\text{s} = 8 \text{ ms}/512$ rows. This method is preferable to burst refresh in most systems.



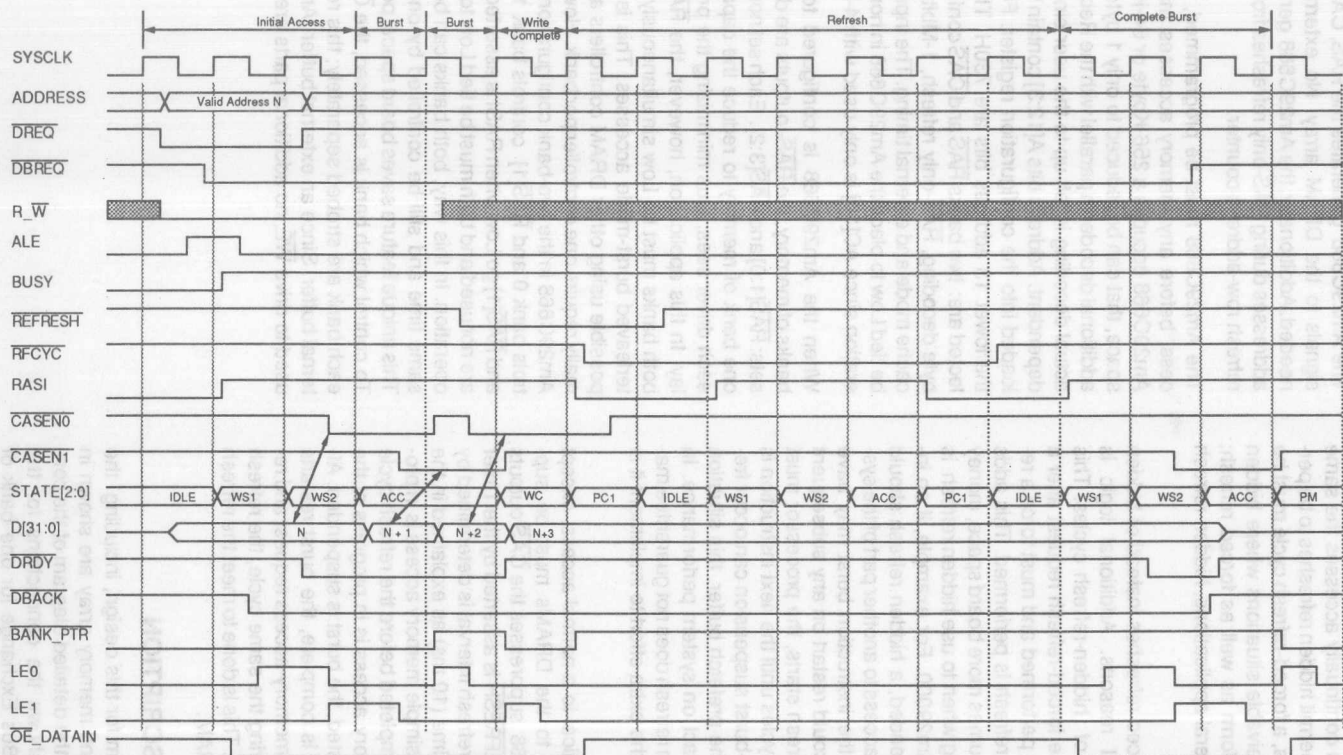
11803-003A

Figure 3a. Instruction Burst Access



11803-004A

Figure 3b. Instruction Burst Access Pre-empted by Data Access



11803-005A

Figure 4. Burst Data Write and Refresh

Hidden refresh has the lowest system impact since all or most of the refresh cycle is overlapped with an access to another memory or I/O device. There are times, however, when the system continually accesses the same memory and does not permit hidden refreshes to be performed. If this happens, a forced-refresh cycle must be used. There are conceivable situations where hidden refresh would not perform as well as forced refresh; however, for most general applications, hidden refresh is the best choice.

This design utilizes forced refreshes instead of hidden refreshes, for several reasons. Additional logic is needed to keep track of hidden-refresh cycles. This logic must suppress the forced-refresh request after a hidden-refresh cycle is performed and must force a refresh when no hidden refresh is performed. This adds extra devices and consumes more board space, money and power. Determining when to use hidden refresh is also difficult with the Am29000. For example, if an instruction burst is suspended, a hidden refresh should not occur due to a data access to another part of the system. This is because the instruction burst may have been suspended and could restart on any subsequent cycle. If the hidden refresh starts, the processor must wait a minimum of 11 cycles until the next instruction is read. Since instruction-burst suspension can occur frequently, due to filling the prefetch buffer, this situation can have a major impact on system performance. In these situations, hidden refresh does not guarantee major savings, therefore, the extra effort to implement it is not justified.

A refresh cycle is identical to a normal access, except that the $\overline{\text{CAS}}$ outputs to the DRAMs must be suppressed. The Am29C668 suppresses the $\overline{\text{CAS}}$ outputs in the refresh mode. REFRESH is asserted by the Timer PAL every 9.8 μs . The refresh interval is determined by maximum $\overline{\text{CAS}}$ active time (10 ms) as explained in the *Timer PAL* section. If a simple memory access is in progress, the access is completed before the refresh cycle begins. If a burst memory access is in progress at the time a refresh is requested, the burst is suspended. After the refresh access is complete, the burst restarts automatically. If both a memory-access request and refresh request occur during the same cycle, the refresh request is given priority. This is done to meet the refresh requirements of the DRAM.

FUNCTIONAL DESCRIPTION

The main block diagram for this design, including the control logic, buffers and memory array are shown in Figure 5a. Figure 5b is the detailed diagram of the control logic. Figure 5c shows the connections for the Am29C983A Multiple Bus Exchange for one bank of memory.

Am29C668 Configurable Dynamic Memory Controller (CDMC)

The Am29C668 generates the $\overline{\text{RAS}}$, $\overline{\text{CAS}}$ and address signals to the DRAM array. No external drivers are needed. Additionally, the Am29C668 generates the row addresses during RAS-only refreshes from the internal-refresh row-address counter.

The Am29C668 must be programmed, via an I/O access, before any memory accesses may occur. The Am29C668 occupies a 256-Kbyte or 64-Kword address space, that can be reduced to only 1 byte by adding an additional decoder in parallel with the Request PAL. The actual decoding is left up to the user since it is system dependent. Address bits A[12:3] contain the value to be loaded into the configuration register. For this design, the lower 13 address bits are 730H. The options selected are: two banks $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$ configuration, $\overline{\text{CAS}}$ byte decoding, RAS-only refresh, 1-Mbit memory size, cache mode and external timing. The input AC[10] must be tied Low to place the Am29C668 in normal-mode operation since AC[10] is only used with 4-Mbit DRAMs.

When the Am29C668 is configured to support two banks of memory, the $\overline{\text{RAS}}_n$ outputs are divided into two sets: $\overline{\text{RAS}}[1:0]$ and $\overline{\text{RAS}}[3:2]$. Each set normally controls one bank of memory to reduce the capacitive loading each driver sees, thus minimizing the propagation delay. In this application, however, the $\overline{\text{RAS}}_n$ outputs of both banks must be Low simultaneously to support interleaved burst-mode accesses. This is generally not possible using other DRAM controllers and would normally require one controller per bank. However, with the Am29C668 in the two-bank configuration, $\overline{\text{RAS}}[0]$ controls bank 0 and $\overline{\text{RAS}}[1]$ controls bank 1. Both $\overline{\text{RAS}}[0]$ and $\overline{\text{RAS}}[1]$ go Low when RASi is asserted; the SEL[1:0] are not used and both must be tied Low to insure proper operation. In this way, both banks can be active at the same time and still be controlled by one Am29C668. This unique feature saves board space, power and cost. To control which bank is accessed, the $\overline{\text{CAS}}_n$ inputs to each bank are strobed separately; this requires an external buffer. Since an external buffer (Am2966) is also used to drive $\overline{\text{WE}}$, no additional parts are required.

11803-001A

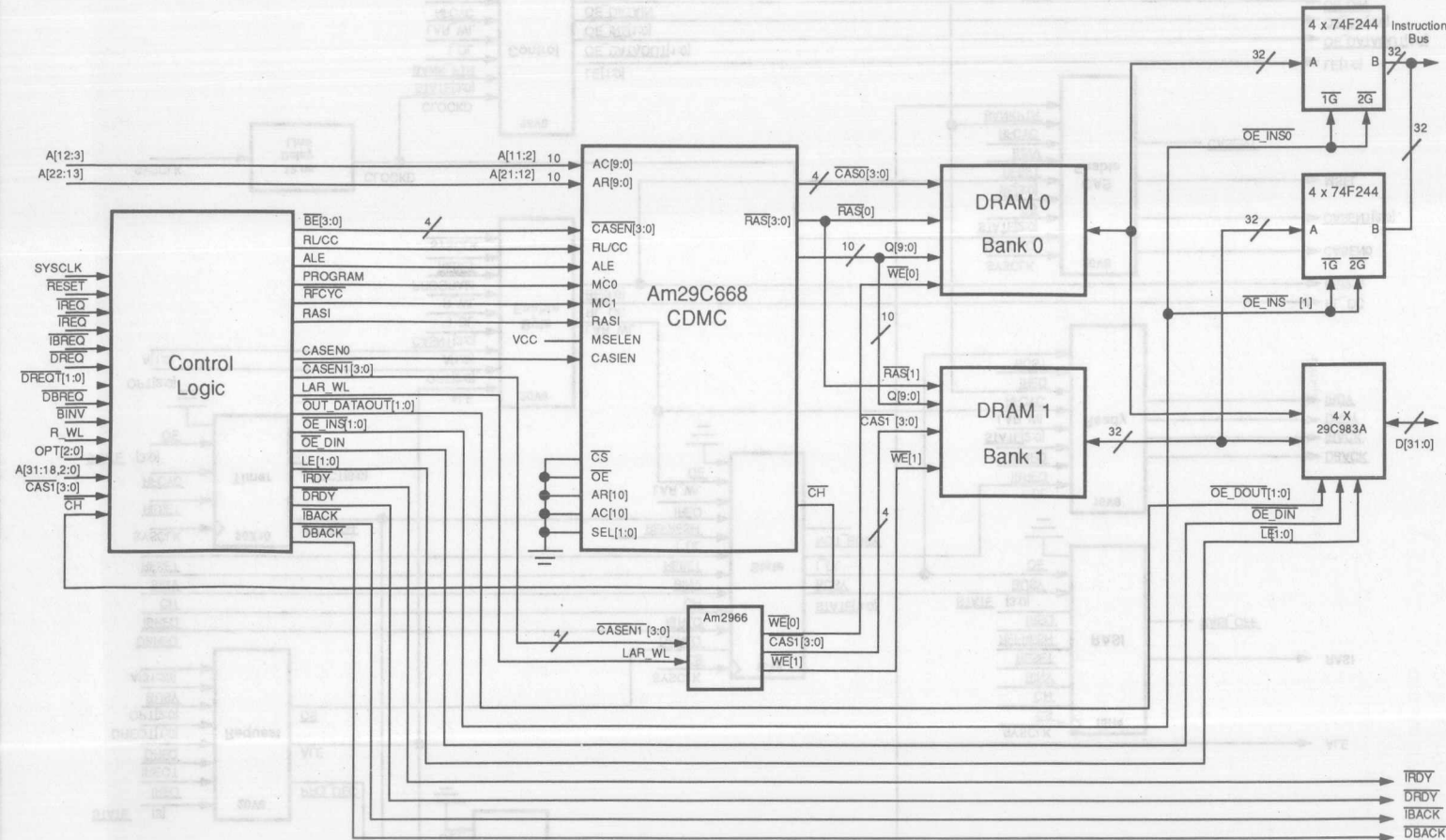


Figure 5a. Top Level Schematic

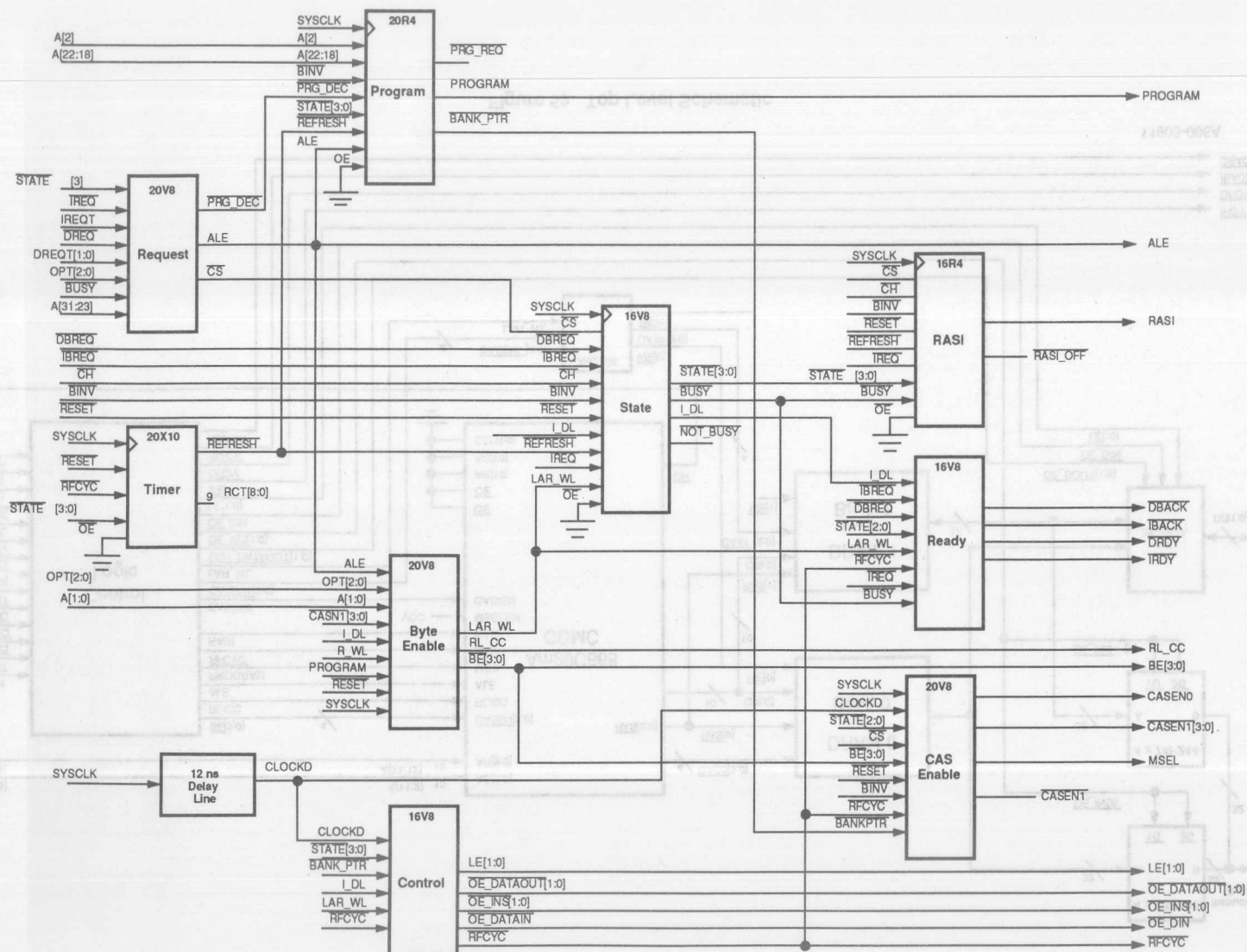


Figure 5b. Control Logic Block Diagram

11803-007A

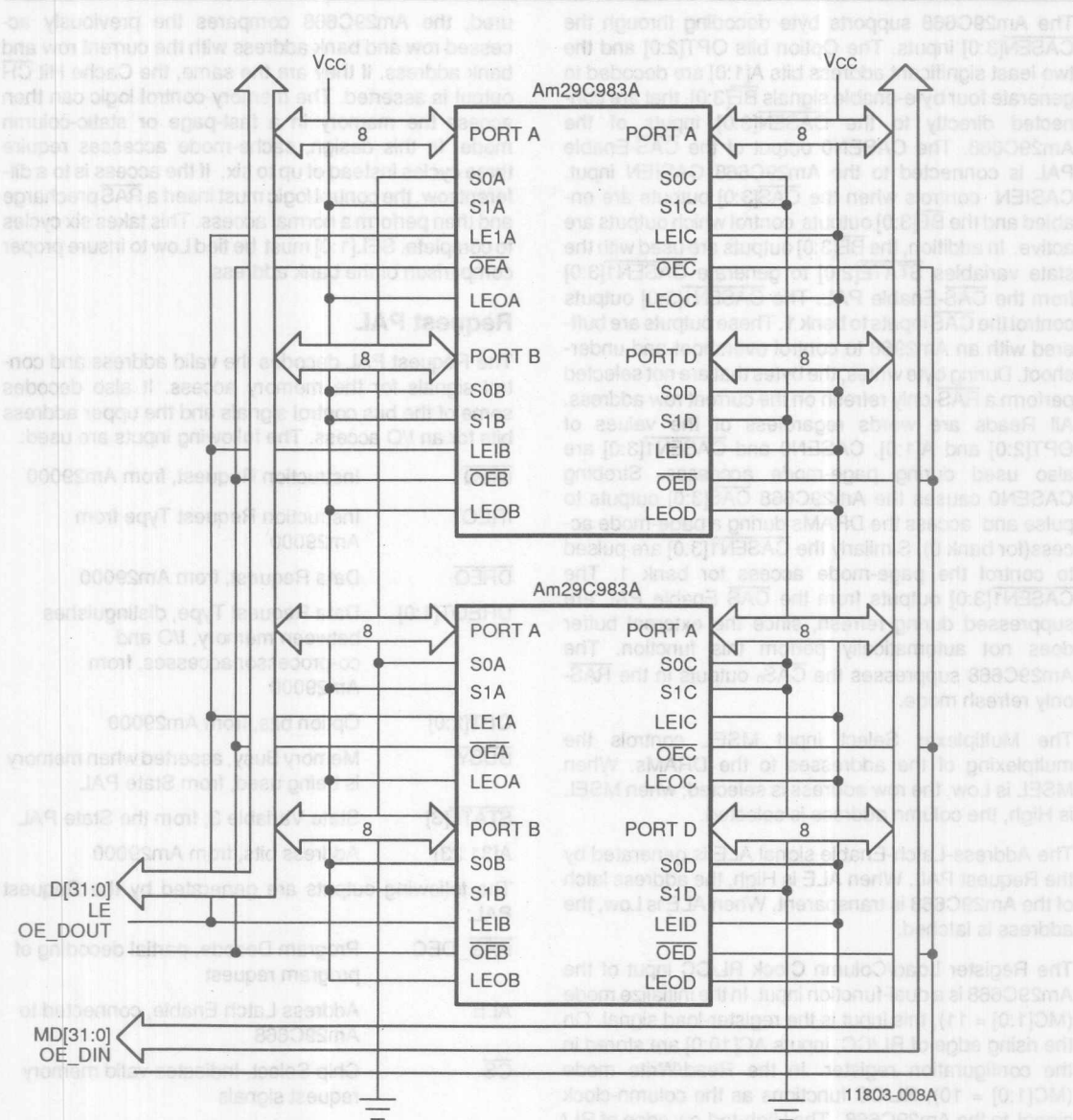


Figure 5c. MBE Connections

The Am29C668 supports byte decoding through the $\overline{\text{CASEN}}[3:0]$ inputs. The Option bits $\text{OPT}[2:0]$ and the two least significant address bits $\text{A}[1:0]$ are decoded to generate four byte-enable signals $\overline{\text{BE}}[3:0]$, that are connected directly to the $\overline{\text{CASEN}}[3:0]$ inputs of the Am29C668. The $\overline{\text{CASEN}}_0$ output of the $\overline{\text{CAS}}$ -Enable PAL is connected to the Am29C668 $\overline{\text{CASEN}}$ input. $\overline{\text{CASEN}}$ controls when the $\overline{\text{CAS}}[3:0]$ outputs are enabled and the $\overline{\text{BE}}[3:0]$ outputs control which outputs are active. In addition, the $\overline{\text{BE}}[3:0]$ outputs are used with the state variables $\text{STATE}[2:0]$ to generate $\overline{\text{CASEN}}_1[3:0]$ from the $\overline{\text{CAS}}$ -Enable PAL. The $\overline{\text{CASEN}}_1[3:0]$ outputs control the $\overline{\text{CAS}}$ inputs to bank 1. These outputs are buffered with an Am2966 to control overshoot and undershoot. During byte writes, the bytes that are not selected perform a $\overline{\text{RAS}}$ -only refresh on the current row address. All Reads are words regardless of the values of $\text{OPT}[2:0]$ and $\text{A}[1:0]$. $\overline{\text{CASEN}}_0$ and $\overline{\text{CASEN}}_1[3:0]$ are also used during page-mode accesses. Strobing $\overline{\text{CASEN}}_0$ causes the Am29C668 $\overline{\text{CAS}}[3:0]$ outputs to pulse and access the DRAMs during a page-mode access (for bank 0). Similarly the $\overline{\text{CASEN}}_1[3:0]$ are pulsed to control the page-mode access for bank 1. The $\overline{\text{CASEN}}_1[3:0]$ outputs from the $\overline{\text{CAS}}$ Enable PAL are suppressed during refresh, since the external buffer does not automatically perform this function. The Am29C668 suppresses the $\overline{\text{CAS}}_n$ outputs in the $\overline{\text{RAS}}$ -only refresh mode.

The Multiplexer Select input MSEL controls the multiplexing of the addresses to the DRAMs. When MSEL is Low, the row address is selected; when MSEL is High, the column address is selected.

The Address-Latch-Enable signal ALE is generated by the Request PAL. When ALE is High, the address latch of the Am29C668 is transparent. When ALE is Low, the address is latched.

The Register Load/Column Clock RL/CC input of the Am29C668 is a dual-function input. In the initialize mode ($\text{MC}[1:0] = 11$), this input is the register-load signal. On the rising edge of RL/CC , inputs $\text{AC}[10:0]$ are stored in the configuration register. In the Read/Write mode ($\text{MC}[1:0] = 10$), RL/CC functions as the column-clock signal to the Am29C668. The High-to-Low edge of RL/CC increments the column counter. ALE must be Low for the counter to increment. If ALE is High, the latch is transparent and the counter does not function properly. The counter function is used during burst-mode accesses. Since both memory banks share the same address bus, the address cannot be incremented until after the $\overline{\text{CAS}}_n$ outputs to bank 1 are asserted. The four $\overline{\text{CAS}}$ outputs to bank 1 are ORed together in the Byte-Enable PAL to generate the RL/CC input to the Am29C668. By using the $\overline{\text{CAS}}$ outputs, the Column-Address-to- $\overline{\text{CAS}}$ hold time is guaranteed.

The Am29C668 has on-chip comparators for implementing cache-mode operation. When properly config-

ured, the Am29C668 compares the previously accessed row and bank address with the current row and bank address. If they are the same, the Cache Hit $\overline{\text{CH}}$ output is asserted. The memory-control logic can then access the memory in a fast-page or static-column mode. In this design, cache-mode accesses require three cycles instead of up to six. If the access is to a different row, the control logic must insert a $\overline{\text{RAS}}$ precharge and then perform a normal access. This takes six cycles to complete. $\text{SEL}[1:0]$ must be tied Low to insure proper comparison of the bank address.

Request PAL

The Request PAL decodes the valid address and control signals for the memory access. It also decodes some of the bus control signals and the upper address bits for an I/O access. The following inputs are used:

$\overline{\text{TREQ}}$	Instruction Request, from Am29000
$\overline{\text{IREQT}}$	Instruction Request Type from Am29000
$\overline{\text{DREQ}}$	Data Request, from Am29000
$\overline{\text{DREQT}}[1:0]$	Data Request Type, distinguishes between memory, I/O and co-processor accesses, from Am29000
$\text{OPT}[2:0]$	Option bits, from Am29000
$\overline{\text{BUSY}}$	Memory Busy, asserted when memory is being used, from State PAL
$\text{STATE}[3]$	State Variable 3, from the State PAL
$\text{A}[31:23]$	Address bits, from Am29000

The following outputs are generated by the Request PAL:

$\overline{\text{PRG_DEC}}$	Program Decode, partial decoding of program request
ALE	Address Latch Enable, connected to Am29C668
$\overline{\text{CS}}$	Chip Select, Indicates valid memory request signals

The Request PAL's main function is to decode a memory access. The Chip Select $\overline{\text{CS}}$ is generated when a valid memory request is generated by the Am29000. The memory occupies the same address space for both data and instruction memory. This is not required and may be changed so that the instruction and data memory occupy different address spaces. For instruction memory accesses, $\overline{\text{IREQT}}$ is decoded. $\overline{\text{IREQT}}$ distinguishes between ROM and RAM accesses. In this implementation, ROM and RAM can occupy the same address space. The 32 address bits are decoded, which provides for a full 4-Gbyte address space.

For data-memory accesses, DREQT[1:0] and OPT[2:0] are decoded to determine the type of memory access (see Tables 2a and 2b). By decoding DREQT[1:0] and OPT[2:0], compatibility with the ADAPT29K is maintained. Also, the memory does not falsely interpret accesses to I/O devices or the coprocessor as memory accesses. The \overline{CS} output does not decode the \overline{BINV} signal. The State PAL uses both \overline{CS} and \overline{BINV} to determine if the memory request is valid.

Table 2a.
Decoding of DREQT[1:0]

DREQT[1:0]	Meaning
00	Instruction/Data-Memory Access
01	Input/Output Access
1X	Coprocessor Transfer

Table 2b.
Decoding of OPT[2:0] Based on DREQT[1:0]

DREQT[1:0]	OPT[2:0]	Meaning
0X	000	Word Length Access
0X	001	Byte Access
0X	010	Half-Word Access
XX	011	Reserved
00	100	Instruction ROM Access (as data)
00	101	Cache Control
00	110	ADAPT29K Accesses
XX	111	Reserved

The ALE signal from the Request PAL is connected to the Am29C668 ALE input. When ALE is High, the Am29C668 address latch is transparent. The Low-going edge of ALE latches the value in the address latch. In most implementations, ALE is \overline{CS} inverted. However, this is not possible in this case, since ALE must be generated 28 ns, at most, after the rising edge of the SYLCLK. For a 20-MHz Am29000, the control signals require a maximum of 16 ns to be valid leaving only 12 ns to generate ALE. If ALE were \overline{CS} inverted, it would require two PAL delays to generate, or 15 ns minimum; therefore, ALE cannot be \overline{CS} inverted. Directly decoding ALE is not feasible either because of the number of address bits that must be decoded. Thus, in this design, ALE is \overline{BUSY} inverted plus an additional term to provide for an instruction-burst restart. \overline{BUSY} is generated by the State PAL and indicates when the memory is being accessed. The additional term is necessary because the memory system supports instruction-burst restart. At the end of an instruction burst, the memory controller

enters the page-mode state and waits for the burst to restart or for a new instruction request. ALE must be held Low in this state to insure that the instruction burst can restart. If a new instruction request is received, ALE must be active in that cycle. Since \overline{BUSY} is still asserted, there must be an additional term that so that ALE can be asserted when \overline{BUSY} is asserted. $\overline{STATE}[3]$ indicates that the memory is in the page-mode state. If \overline{IREQ} is asserted during this state, ALE is also asserted so that the new instruction can begin without delay.

Program Decode $\overline{PRG_DEC}$ from the Request PAL generates a partial decoding of a programming request, because of the large number of address bits that must be decoded. Since the configuration register occupies a 256 Kbyte block, A[31:18] must be decoded. This cannot be done simply and efficiently in a single PAL. By using a two-stage decoder, Request and Program PALs, the programming requests are easily decoded.

Program PAL

The Program PAL is used to decode the lower address bits and $\overline{REFRESH}$ signal for an I/O access to load the Am29C668 configuration register. This PAL also maintains the Bank-Pointer signal $\overline{BANK_PTR}$ that determines which bank is currently active. The following inputs are used:

SYSCLK	20 MHz System Clock
$\overline{STATE}[3:0]$	State Variables, from State PAL
A[2]	Address Bit 2, from Am29000
A[22:18]	Address, from Am29000
\overline{BINV}	Bus Invalid, from Am29000
ALE	Address Latch Enable, from Request PAL

$\overline{PRG_DEC}$	Program Decode, from Request PAL
$\overline{REFRESH}$	Refresh Request, from Timer PAL

The following outputs are generated:

$\overline{PRG_REQ}$	Program Request, Signals Valid Decoding of Program Request
PROGRAM	Program, Connects to Am29C668 MC0 Input
$\overline{BANK_PTR}$	Bank Pointer, Indicates Currently Active Bank

The $\overline{PRG_REQ}$ signal is a combinatorial output and is asserted when A[22:18] are the same as the addresses of the configuration register. Currently the Am29C668 occupies 256 Kbyte of I/O address space. This can be reduced to only one byte by adding an address decoder in parallel with the Request PAL. This decoder would decode the lower 18 address bits A[17:0]. By using this parallel decoding scheme to detect valid I/O requests,

the full 32-bit address is decoded and all of the 4-Gbyte I/O address space can be used.

PROGRAM is a registered output, asserted if PRG_REQ and PRG_DEC are valid and BINV and REFRESH are deasserted. It is connected to the Am29C668 MC0 input. When MC1 and MC0 are both one, the Am29C668 configuration register is loaded on the rising edge of the register load RL/CC signal, generated by the Byte-Enable PAL. Programming requires two clocks as shown in Figure 6. The programming logic does not distinguish between I/O Reads and Writes and configures the Am29C668 CDMC on either valid access. REFRESH must be deasserted so that the State PAL does not assert RFCYC, the MC1 input to the Am29C668.

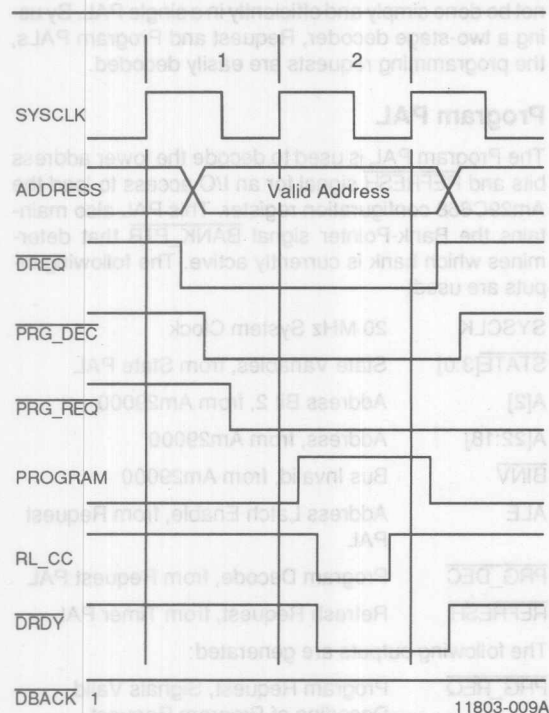


Figure 6. Program Cycle

The bank-pointer signal $\overline{\text{BANK_PTR}}$ indicates the currently active bank, which is initially determined by the value of A[2] during the initial access. If A[2] is 0, bank 0 is active; if A[2] is 1, bank 1 is active. A[2] is used because consecutive words must be in different banks for burst accesses to occur with no wait states. $\overline{\text{BANK_PTR}}$ is toggled in the access state.

Byte-Enable PAL

This PAL generates the $\overline{\text{BE}}$ and latched-Read/Write signals used during data-memory accesses. It generates the register RL/CC signal used during programming and burst-mode accesses. The following inputs are used:

ALE	Address Latch Enable, from Request PAL
OPT[2:0]	Option, from Am29000
A[1:0]	Least Significant Address Bits 1 and 0, from Am29000
CAS[3:0]	CAS Outputs to Bank 1, from CAS-Enable PAL
I_DL	Instruction Active High, Data Active Low, from Request PAL
R_WL	Read Active High, Write Active Low, from AM29000
PROGRAM	Program, from Program PAL
RESET	System Reset Signal
SYSCLK	20-MHz System Clock
RL/CC	Register Load/Column Clock
LAR_WL	Latched Read Active High, Write Active Low
$\overline{\text{BE}}[3:0]$	Byte Enable

The RL/CC signal is a dual-function output. During programming cycles, it loads the Am29C668 configuration register on its rising edge. During memory accesses, it increments the Am29C668 column counter on its falling edge. This function is used during burst accesses to increment the column address.

The Byte-Enable PAL latches the Read/Write R_WL signal generated by the Am29000. The latched signal is LAR_WL. When ALE is High, the latch is transparent; and when ALE is Low, the value is latched. R_WL is undefined during instruction accesses. LAR_WL is forced High during instruction accesses because all instruction accesses are Read cycles.

The byte-enable outputs $\overline{\text{BE}}[3:0]$ are generated by decoding the option bits OPT[2:0] and the two least significant address bits A[1:0]. See Table 3. The Am29000 supports both Big Endian and Little Endian addressing. Big Endian numbers the bytes and half words from the most significant bit to the least significant bit, while Little Endian numbers from the least significant to the most significant. This design assumes Big Endian, as does all current 29K software. If a Little Endian implementation is required, the variable BYTEORDER must be changed from 0 to 1 in the PAL equations.

Table 3. Byte-Enable Decoding

OPT[2:0]	A[1:0]	BE[3:0]	Access Type
000	XX	0000	Word
001	00	0111	Byte 0
001	01	1011	Byte 1
001	10	1101	Byte 2
001	11	1110	Byte 3
010	00	0011	Half Word 0
010	10	1100	Half Word 1

Note: Big Endian assumed.

CAS-Enable PAL

This PAL generates the CAS-Enable signals that control the CAS outputs to the memory. The following inputs are used:

SYSCLK	20-MHz System Clock
CLOCKD	Delayed SYSCLK
STATE[2:0]	State Variables, from State PAL Indicates Current Memory Cycle
BE[3:0]	Byte Enable, from the Byte-Enable PAL
CS	Chip Select, from Request PAL
RFCYC	Refresh Cycle, from State PAL
BANK_PTR	Bank Pointer, from Control PAL Indicates Which Bank is Currently Active
RESET	System-Reset Signal
BINV	Bus Invalid, from Am29000

The following outputs are generated:

CASEN0	CAS Enable for Bank 0
CASEN1[3:0]	CAS Enable for Bank 1
CASEN1	Used to Determine Which Bank has Completed its Access.
MSEL	Multiplexer Select, Connected to the Am29C668 MSEL Input.

CASEN0 is the input to the Am29C668 CASIEN. When CASEN0 is asserted, the CAS[3:0] outputs of the Am29C668 are enabled. Since the auto-timing is used, the CASIEN input is an external override for controlling the CAS outputs.

The CASEN1[3:0] outputs are the inputs to the Am2966 external buffer and control the CAS inputs of bank 1. Each output controls one byte. During refresh cycles, the CASEN1[3:0] outputs are always deasserted. The Am29C668 deasserts the CAS outputs during refresh cycles, but the external buffer's CAS outputs follow the

inputs. Therefore all the CASEN1[3:0] outputs are suppressed.

CASEN1 is used to determine if CASEN0 or CASEN1[3:0] is deasserted on the rising edge of SYSCLK. This assures maximum CAS precharge time since the CASEN0 and CASEN1[3:0] outputs turn off on the rising edge of SYSCLK. If this were not done, some of the CAS precharge time would be lost waiting for BANK_PTR and STATE[3:0].

MSEL controls address multiplexing to the DRAMs. When MSEL is Low, the row address is selected, when MSEL is High, the column address is selected.

Control PAL

The Control PAL generates the signals to enable and disable the output drivers and control the data latches. The following inputs are used:

SYSCLK	20-MHz System Clock
STATE[3:0]	State Variables, from State PAL
I_DL	Instruction or Data Cycle Indicator, from Request PAL
LAR_WL	Latched Read or Write Cycle Indicator, from Byte-Enable PAL
REFRESH	Refresh Request, from Timer PAL
BANK_PTR	Bank Pointer, from Program PAL

The following outputs are generated:

OE_INS[1:0]	Output Enable for Instruction-Bus Buffers
OE_DOUT[1:0]	Output Enable for Data-Bus Buffers
OE_DIN	Output Enable Data Input
LE[1:0]	Latch Enables for Data-Bus Latches
RFCYC	Refresh Cycle, Connected to Am29C668 MC1 Input and Used to Indicate That a Refresh Cycle is in Progress

The memory connects to two separate data and instruction buses. While the instruction bus is Read only, the data bus is bidirectional and four Am29C983A Multiple Bus Exchangers (MBE) are used as interface. By using the input latches in the MBE, single-cycle Write cycles can be performed. The data is removed from the bus one cycle before it is written to memory; thus two Write cycles can be overlapped. Since the instruction bus is Read-only, simple 74F244 buffers are used for interfacing.

The OE_INS[1:0] outputs enable the output drivers of the instruction bus. The instruction buffers are only turned on when the memory is in the access state and the bank is active as indicated by BANK_PTR. RFCYC must also be deasserted to insure that the buffers do not

drive the bus during refresh cycles. The buffers do not turn on until at least two cycles after the access is initiated, providing ample time for the previous device controlling the bus to get off.

The $\overline{OE_DOUT}[1:0]$ outputs perform the same function as the $\overline{OE_INS}[1:0]$ outputs. The data drivers are only turned on when the memory is in the access state, the bank is active as indicated by $\overline{BANK_PTR}$ and \overline{RFCYC} is deasserted. Similarly, the data buffers do not turn on until at least two cycles after the access is initiated.

The $\overline{OE_DIN}$ input controls the outputs driving the data to both banks of DRAMs. The drivers are turned on as early as possible to insure the data set-up time is guaranteed.

The $\overline{LE}[1:0]$ outputs control the input data latches. The data from the Am29000 is valid 20 ns after the rising edge of the system clock and is held for 4 ns after the next rising edge of the system clock. The \overline{LE} outputs of this PAL cannot be generated synchronously from \overline{SYSCLK} because, in the worst case, \overline{LE} would be deasserted 4 ns after data is invalid, violating the Am29C983A setup and hold times. If a synchronous design were used, a faster clock signal would be needed. Since the system clock is already 20 MHz, the logical choice is a 40-MHz clock signal, twice the \overline{SYSCLK} frequency. This design would be difficult because the skew between the two clock signals must be carefully controlled. A simpler and therefore better design is to use a delay line to generate a delayed clock signal \overline{CLOCKD} , so that $\overline{LE}[1:0]$ are combinatorial outputs that are valid only while \overline{CLOCKD} is High. \overline{SYSCLK} is delayed long enough to insure that $\overline{BANK_PTR}$ is valid and the $\overline{LE}[1:0]$ outputs are not falsely asserted.

The refresh cycle output \overline{RFCYC} , which is only asserted during refresh cycles, is connected to the Am29C668 MC1 input. When \overline{RFCYC} is asserted, the Am29C668 drives the current refresh row address on its $\overline{Q}[10:0]$ outputs. The internal counter is updated at the end of each refresh cycle to insure that all rows are refreshed. \overline{RFCYC} is asserted during the IDLE cycle to guarantee the MC1-to-RASI set-up time. \overline{RFCYC} is deasserted during the PC1 cycle to meet the MC1-to-RASI hold time.

Ready PAL

This PAL generates the ready and burst acknowledgement signals for both instruction and data accesses. The following inputs are used:

$\overline{I_DL}$	Instruction or Data Indicator, from Request PAL
\overline{IREQ}	Instruction Request from Am29000
\overline{IBREQ}	Instruction Burst Request, from Am29000
\overline{DBREQ}	Data Burst Request, from Am29000

$\overline{STATE}[2:0]$	State Variables, from State PAL
\overline{CS}	Chip Select, from Request PAL
$\overline{LAR_WL}$	Latched Read/Write Signal, from Byte Enable PAL
\overline{RFCYC}	Refresh Cycle Indicator, from State PAL
\overline{BUSY}	Busy from State PAL, Indicates When Memory is Accessed

The following outputs are generated:

\overline{IBACK}	Instruction Burst Acknowledge
\overline{DBACK}	Data Burst Acknowledge
\overline{IRDY}	Instruction Ready, Indicates Valid Instruction on the Instruction bus
\overline{DRDY}	Data Ready, Indicates Valid Data on the Data Bus

The \overline{DRDY} and \overline{DBACK} outputs are used to inform the processor about the status of data-memory accesses. \overline{DRDY} is asserted at the completion of data Read or Write cycles. \overline{DBACK} is asserted when the memory supports burst-mode accesses. When \overline{DBACK} is asserted, the Am29000 is freed to start an instruction access if one is pending. Therefore, all the address and control lines needed by the memory in subsequent cycles must be latched. This is insured by not asserting \overline{DBACK} until \overline{ALE} is deasserted. \overline{DBACK} is not deasserted as the result of a refresh request. If a burst access is currently in progress, it is completed. The burst is delayed by not asserting \overline{DRDY} . The refresh cycle is performed and then the burst is resumed. If this were not done, a refresh request would terminate the burst access. The processor would then have to try to restart the burst access, causing more contention for the address bus. \overline{DRDY} is also asserted during any cycle when PROGRAM is active.

The \overline{IRDY} and \overline{IBACK} outputs function the same as \overline{DRDY} and \overline{DBACK} except they are used for instruction cycles rather than data cycles. Because the memory supports instruction burst re-start, \overline{IBACK} must be asserted as long as the memory can restart the suspended burst. \overline{IBACK} is latched until a new cycle is started when the Am29000 asserts \overline{IREQ} . \overline{IBACK} is deasserted as soon as \overline{IREQ} is asserted, preventing \overline{IBACK} from being falsely asserted.

To support multiple devices, the same control signals from different devices are externally OR-ed together to obtain the appropriate control signals to the Am29000. It is difficult to implement the transfer of control by selectively driving the control lines with three-state buffers as is commonly done in slower memory systems. Wire OR-ing with open-collector drivers is similarly impractical. Using an SSI gate or PAL is the only practical method to support multiple devices on the same bus.

Timer PAL

The Timer PAL generates the refresh requests. The following inputs are used:

SYSCLK	20-MHz System Clock
RESET	System Reset; Initializes Counter
RFCYC	Refresh Request; Signals Refresh-Memory Access
STATE[3:0]	State Variables, from State PAL

The following outputs are used:

RCT[8:0]	Counter
REFRESH	Forced Refresh Request

The DRAMs have a maximum $\overline{\text{CAS}}$ active time of 10 ms. If $\overline{\text{CAS}}$ is active longer than 10 ms, the memory could be corrupted; therefore, the refresh interval should be less than 10 ms to ensure that the maximum $\overline{\text{CAS}}$ active time is not violated. If the system can always guarantee that $\overline{\text{CAS}}$ is not active longer than 10 ms, the refresh interval can be extended to 15.6 μs . This reduces the refresh overhead from 3.5 to 2.2%.

The Timer PAL helps implement the forced refreshes along with the State PAL. The period for the Timer PAL is selected by the value initialized in the counter. This value is set to 195 resulting in a refresh request cycle time of 9.8 μs = 196 x 50 ns (an extra cycle is included since the counter decrements to 0 before resetting). The initial value of the count is determined by the location of INIT in the logic equations (see the PAL Equations section). REFRESH is asserted when the counter decrements to zero and until the memory finishes the memory refresh. This is indicated by RFCYC asserted and the memory in the access state.

There are several alternate methods to implement the refresh timer. A 555 timer could be used and would require less board space and cost less; however, this solution requires asynchronous arbitration. Another alternative is to use a spare DMA channel to implement the refresh requests similar to the PC-AT* and PS/2* systems. This is practical only if there are spare DMA channels available.

RASI PAL

This PAL generates the Row Address Strobe Input (RASI) for the Am29C668, that causes the appropriate RAS_n output to be asserted and start the internal timing chain. The following inputs are used:

SYSCLK	20-MHz System Clock
STATE[3:0]	State Variables, from State PAL
BUSY	Busy, from State PAL
REFRESH	Refresh Request, from Timer PAL
BINV	Bus Invalid, from Am29000
RESET	System Reset Signal
CS	Chip Select, from Request PAL
CH	Cache Hit Signal, from Am29C668
IREQ	Instruction Request, from Am29000

The following outputs are generated:

RASI_OFF	Indicates Certain Conditions where RASI Must be Deasserted
RASI	Row Address Strobe Input, Connected to Am29C668 RASI input.

The RASI output is connected to the Am29C668 RASI input and is used to control the $\overline{\text{RAS}}_n$ outputs and to start all memory accesses, Read/Write and refresh cycles. Once RASI is asserted, it remains asserted until RASI_OFF is asserted. RASI_OFF is required because there were not enough product terms to implement RASI directly. Keeping RASI asserted provides for fast-page-mode accesses. The $\text{RAS}[0]$ output is the only Am29C668 $\overline{\text{RAS}}$ output used in this design; it is generated by inverting RASI.

State PAL

The State PAL is responsible for arbitrating between memory accesses. It also implements the memory state machine. The following inputs are used:

SYSCLK	20-MHz System Clock
CS	Chip Select, from Request PAL
DBREQ	Data Burst Request, from Am29000
IBREQ	Instruction Burst Request, from Am29000
IREQ	Instruction Request, from Am29000
BINV	Bus Invalid, from Am29000
LAR_WL	Latched Read or Write Signals, from Byte Enable PAL
CH	Cache Hit, from Am29C668
RESET	System Reset
REFRESH	Refresh, from Timer PAL

The following outputs are generated:

STATE[3:0]	State Variables, Indicate Current Memory State
BUSY	Busy, Asserted When Memory is Being Accessed
NOT_BUSY	Not Busy, Used to Indicate When BUSY Should be Deasserted
I_DL	Instruction Cycle Active High, Data Cycle Active Low

The **I_DL** output is used to indicate the type of memory access currently being performed. When **I_DL** is High, an instruction cycle is being performed; when it is Low, a data cycle is performed. **I_DL** is also Low when an instruction access is initiated outside of this modules address space so that the control logic will not mistakenly restart an instruction burst.

This PAL controls the memory. Figure 7 shows the memory state diagram. There are seven memory states:

IDLE	Idle State or Arbitration
WS1	Wait State One
WS2	Wait State Two
ACC	Memory Access
WC	Write Complete
PC1	Pre-charge State One
PM	Page Mode

The state machine uses four state variables **STATE[3:0]**. The lower three state variables **STATE[2:0]** can be used by the other PALs to determine the present state of the memory. This is possible if the PAL does not need to distinguish between the **IDLE** and **PM** states as they have the same lower three bits. This is possible in the Ready PAL for instance.

The memory always starts in the **IDLE** state (Figure 7). On **RESET**, the state machine goes to **IDLE** and remains there until a refresh request or memory-access request. When a valid memory request is generated by the Am29000, the state machine goes from the **IDLE** state to **WS1**. In this state, **RAS** is asserted to start the internal timing chain in the Am29C668. From **WS1**, the state machine goes unconditionally to **WS2**. In the second half of **WS2**, the appropriate **CASEN1[3:0]** or **CASEN0** signals are asserted. This guarantees the **RAS**-to-**CAS** timing. The memory finally goes to the

ACC state, completing the initial memory access. If a burst access is requested and there are no pending memory accesses or refresh accesses, the memory remains in the **ACC** state. When the burst is completed, the state machine unconditionally goes to the **PM** state. In this state, **RAS** is held High and the **CASEN1[3:0]**, **CASEN0** and **BANK_PTR** outputs are not changed. Instruction bursts can be restarted in this state in the cycle after **IBREQ** is asserted. Other accesses may be initiated in only three cycles if they are to the currently active page.

A refresh cycle requested by the timer PAL asserting **REFRESH** is given priority over memory accesses. If a refresh request is received while the memory is accessed, the refresh request waits until the current memory access is completed. The memory state machine precharges **RAS** by going to states **PC1** and **IDLE**. In both these states, **RAS** is held Low, thereby precharging **RAS**. The **RAS**-only refresh cycle follows. The refresh cycle is identical to a memory access, except that **CAS** is suppressed to the DRAMs. Finally another precharge **RAS** cycle occurs. If there are no outstanding memory requests, the state machine remains in the **IDLE** state. If the refresh request occurs during a burst access, the burst access is suspended by the memory-control logic and restarted after the refresh access is complete.

To prevent any possible deadlock, an instruction burst is the only burst access that may be interrupted by another memory request. If a data-memory request is generated while an instruction burst is in progress, the state machine deasserts **BUSY**, thereby asserting the Address Latch Enable **ALE** input. This adds one extra cycle to the data access, which is not critical since this situation only occurs when loading a program into the same memory that the Am29000 is using for execution. This occurs infrequently.

A **BUSY** signal indicates to the control logic that the memory is currently being accessed. It is asserted during data-memory accesses and is not deasserted until the memory access/burst is complete. **BUSY** is also asserted during instruction accesses, but may be deasserted if a data request is pending. **BUSY** is always asserted until the state machine reaches the **ACC** state; the current access must always be completed. The output **NOT_BUSY** is used to indicate when **BUSY** must be deasserted, because it was not possible to implement **BUSY** with the available product terms in a single output.

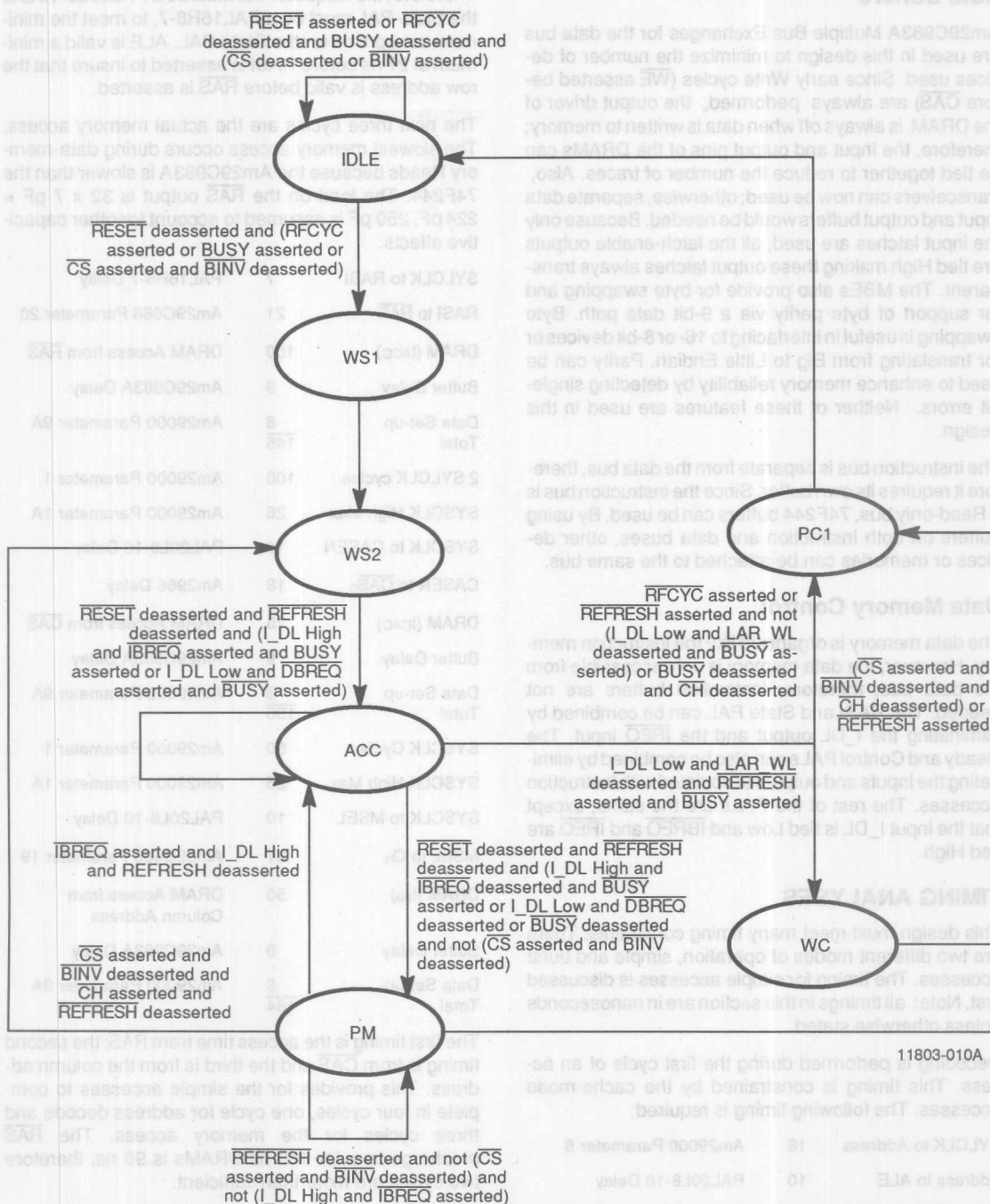


Figure 7. State Diagram

Data Buffers

Am29C983A Multiple Bus Exchanges for the data bus are used in this design to minimize the number of devices used. Since early Write cycles (\overline{WE} asserted before \overline{CAS}) are always performed, the output driver of the DRAM is always off when data is written to memory; therefore, the input and output pins of the DRAMs can be tied together to reduce the number of traces. Also, transceivers can now be used; otherwise, separate data input and output buffers would be needed. Because only the input latches are used, all the latch-enable outputs are tied High making these output latches always transparent. The MBEs also provide for byte swapping and for support of byte parity via a 9-bit data path. Byte swapping is useful in interfacing to 16- or 8-bit devices or for translating from Big to Little Endian. Parity can be used to enhance memory reliability by detecting single-bit errors. Neither of these features are used in this design.

The instruction bus is separate from the data bus, therefore it requires its own buffer. Since the instruction bus is a Read-only bus, 74F244 buffers can be used. By using buffers on both instruction and data buses, other devices or memories can be attached to the same bus.

Data Memory Control

The data memory is organized like the instruction memory. However, the data memory is only accessible from the data bus; therefore, instruction buffers are not needed. The RASi and State PAL can be combined by eliminating the I_DL output and the \overline{IREQ} input. The Ready and Control PALs can also be combined by eliminating the inputs and outputs associated with instruction accesses. The rest of the PALs are the same, except that the input I_DL is tied Low and \overline{IBREQ} and \overline{IREQ} are tied High.

TIMING ANALYSES

This design must meet many timing constraints. There are two different modes of operation, simple and burst accesses. The timing for simple accesses is discussed first. Note: all timings in this section are in nanoseconds unless otherwise stated.

Decoding is performed during the first cycle of an access. This timing is constrained by the cache-mode accesses. The following timing is required:

SYLCLK to Address	16	Am29000 Parameter 6
Address to ALE	10	PAL20L8-10 Delay
ALE to \overline{CH}	16	Am29C668 Parameter 34
\overline{CH} Set-up	7	PAL16R6-7 tsu
Total	49	

Therefore, the Request PAL must be a PAL20L8-10 and the State PAL must be a PAL16R8-7, to meet the minimum set-up time for the State PAL. ALE is valid a minimum of 27 ns before RASi is asserted to insure that the row address is valid before RAS is asserted.

The next three cycles are the actual memory access. The slowest memory access occurs during data-memory Reads because the Am29C983A is slower than the 74F244. The load on the \overline{RAS} output is $32 \times 7 \text{ pF} = 224 \text{ pF}$. 250 pF is assumed to account for other capacitive effects.

SYLCLK to RASi	7	PAL16R6-7 Delay
RASi to \overline{RAS}	21	Am29C668 Parameter 20
DRAM (tACC)	100	DRAM Access from \overline{RAS}
Buffer Delay	9	Am29C983A Delay
Data Set-up	8	Am29000 Parameter 9A
Total	145	
2 SYLCLK cycles	100	Am29000 Parameter 1
SYSCLK High Max	26	Am29000 Parameter 1A
SYSCLK to CASEN	10	PAL20L8-10 Delay
CASEN to \overline{CAS}_n	18	Am2966 Delay
DRAM (tCAC)	25	DRAM Access from \overline{CAS}
Buffer Delay	9	Am29C983A Delay
Data Set-up	8	Am29000 Parameter 9A
Total	196	
SYSCLK Cycle	50	Am29000 Parameter 1
SYSCLK High Max	26	Am29000 Parameter 1A
SYSCLK to MSEL	10	PAL20L8-10 Delay
MSEL to Q_n	31	Am29C668 Parameter 19
DRAM (tAA)	50	DRAM Access from Column Address
Buffer Delay	9	Am29C983A Delay
Data Set-up	8	Am29000 Parameter 9A
Total	184	

The first timing is the access time from \overline{RAS} ; the second timing is from \overline{CAS} and the third is from the column address. This provides for the simple accesses to complete in four cycles, one cycle for address decode and three cycles for the memory access. The \overline{RAS} precharge time for 100 ns DRAMs is 90 ns, therefore two cycles are more than sufficient.

The \overline{IRDY} and \overline{DRDY} signals require the following timing:

SYSCLK to ACC_ST	7	PAL16R6-7 Delay
ACC_ST to DRDY	10	PAL16L8-10 Delay
External NOR gate	7.5	PAL16L8-7 Delay
DRDY Set-up	16	Am29000 Parameter 9B
Total	40.5	

For IBACK and DBACK the following timing applies:

SYSCLK to IREQ	16	Am29000 Parameter 6
IREQ to IBACK	10	PAL16L8-10 Delay
External NOR gate	7.5	PAL16L8-7 Delay
IBACK Set-up	15	Am29000 Parameter 9B
Total	48.5	

This means that a PAL16L8-10 is required for the Ready PAL if the external device that generates the ready and burst acknowledge signals to the Am29000 is a PAL16L8-7.

A burst access must take less than two cycles to meet the single-cycle burst requirement. The load on the outputs of the Am2966 is assumed to be 70 pF.

SYSCLK High Max	26	Am29000 Parameter 1A
SYSCLK to CASEN1	10	PAL20L8-10 Delay
CASEN1 to CAS	18	Am2966 Delay
DRAM (t _{cac})	25	DRAM Access from CAS
Buffer Delay	9	Am29C983A Delay
Data Set-up	8	Am29000 Parameter 9A
Total	96	

During burst accesses, the column address must meet the DRAMs setup time of 0 ns. The timing is for the address is:

SYSCLK High Max	26	Am29000 Parameter 1A
SYSCLK to CASEN1	10	PAL20L8-10 Delay
CASEN1 to CAS	18	2966 Delay
CAS to RL_CC	10	PAL20L8-10 Delay
RL_CC to Q _n	30	Am29C668 Parameter 27
Total	94	

The CAS to bank 0 requires:

One processor cycle Min	50	Am29000 Parameter 1
Pre-Charge Cycle	26	1/2 SYSCLK Cycle
SYSCLK to CASIEN0	10	PAL20L8-10 Delay
CASIEN to CAS	12	Am29C668 Parameter 26
Total	98	

Therefore, the address is valid 4 ns before CAS. This is the difference between the slowest address to the fastest CAS. For CAS precharge during the fast-page-mode accesses, CAS must be deasserted for at least 15 ns for most DRAMs. Some manufactures make DRAMs with CAS-precharge time t_{cp} of 10 ns. Because the Am29C668 has symmetric outputs, the rise and fall times are the same. The outputs of the PAL driving the CASIEN input of the Am29C668 do not have symmetric rise and fall times. The t_{cp}, therefore, is shorter than the minimum clock-High time of the Am29000, 24 ns parameter 1A. For the minimum t_{cp} to be violated, the skew between the rise and fall times of the PAL would have to be greater than 9 ns. This would never be the case, therefore the CAS precharge can be met under worst-case conditions.

For programming cycles, the address must be decode in one cycle:

SYSCLK to Address	16	Am29000 Parameter 6
Address to PRG_REQ	10	PAL16L8-10 Delay
PROGRAM Set up	10	PAL20R4-10 tsu
Total	36	

The delay line used to generate CASEN0, CASEN1[3:0] and LE[1:0] must delay SYSCLK long enough for BANK_PTR and BINV to reach their final values. If this delay is too long, the data buffers can latch the wrong data on Write cycles. Therefore, the Program PAL should be a PAL20R4-10. BANK_PTR is valid 8 ns after the rising edge of SYSCLK and BINV is valid 9 ns after the falling edge of SYSCLK. Therefore, the delay line must be at least 9 ns. Because most delay lines are accurate to ±2 ns, the delay line should be at least 11 ns. The maximum delay is computed as follows:

SYSCLK	50	Am29000 Parameter 1
SYSCLK to Data Hold time	4	Am29000 Parameter 20
SYSCLK High Max	-26	Am29000 Parameter 1A
CLOCKD to LE	-10	PAL20L8-10 tsu
Data Hold Time	-2.5	Am29C983A Parameter 14
Total	15.5	Maximum Delay

Therefore, a nominal value of 12 ns would be best for the delay line and the Control PAL should be a PAL16L8-10.

PARTS LIST

Part	Count
PALCE16V8-10	3
PAL16R4-7	1
PALCE20V8-10	3
PAL20R4-10	1
PAL20X10A	1
Am2966	1
Am29C668	1
74F244	8
Am29C983A	4
Memories	64
12 ns Delay Line	1
Total	88

A discrete memory controller, described in the *Am29000 32-Bit Streamlined Instruction Processor Memory Design Handbook* Chapter 6, requires 113 devices including memory chips. The design, described here, represents a 22.1% reduction in parts. The memory controller in utilizing the Am29C668 requires 12 devices while the discrete design requires 25 or 208 % more. The Am29C668 offers a high degree of integra-

tion, which reduces system cost, power consumption, board space and design time. However, the discrete design has one advantage in that it can operate in a 25 MHz system providing up to a 25% performance improvement. The Am29C668 can also be used in higher speed systems, but the control logic must be changed.

VARIATIONS ON THIS DESIGN

One change to this design is to permit only data-Write accesses to instruction memory. This eliminates the output buffers on the data bus, reducing the required number of parts. The Am29C983As could be replaced with 74F373s to save cost.

This design is limited to 20 MHz due to the control logic. $\overline{\text{IBREQ}}$ and $\overline{\text{DBREQ}}$ are used by the control logic and are valid very late in the cycle. For faster versions of the Am29000, $\overline{\text{IBREQ}}$ and $\overline{\text{DBREQ}}$ do not allow for sufficient setup time to be registered by the PALs. However, the Am29C668 can be used in 33-MHz systems, presently the fastest Am29000, with a different control logic. Two banks of SCDRAMs with one Am29C668 controlling each bank provides maximum performance. Using 70-ns SCDRAMs, the initial cycle requires five cycles and burst accesses require one cycle.

PAL Equations

The following application notes are guides to interfacing the Am29C668 with popular microprocessors. They are paper designs only, and have not been built and tested.

The following are the logic equations for the PALs. They are written in PALASM.

"32-Bit Memory Design for the 29K. This device generates and latches the byte enable signals. Opt[2:0] and A[1:0] are decoded to determine which bytes are being accessed. ALE is used to latch the values."

;PALASM Design Description

----- Declaration Segment -----

TITLE PROGRAM

PATTERN A

REVISION 1.0

COMPANY AMD INC.

DATE 07/12/91

CHIP _PROG29K PAL20R4

----- PIN Declarations -----

PIN 1	SYCLK	COMBINATORIAL; INPUT
PIN 2	/MSTATE[3]	COMBINATORIAL; INPUT
PIN 3	/MSTATE[2]	COMBINATORIAL; INPUT
PIN 4	/MSTATE[1]	COMBINATORIAL; INPUT
PIN 5	/MSTATE[0]	COMBINATORIAL; INPUT
PIN 6	A[2]	COMBINATORIAL; INPUT
PIN 7	ALE	COMBINATORIAL; INPUT
PIN 8	/BINV	COMBINATORIAL; INPUT
PIN 9	/REFRESH	COMBINATORIAL; INPUT
PIN 10	/PRG_DEC	COMBINATORIAL; INPUT
PIN 11	A[22]	COMBINATORIAL; INPUT
PIN 14	A[21]	COMBINATORIAL; INPUT
PIN 21	A[20]	COMBINATORIAL; INPUT
PIN 22	A[19]	COMBINATORIAL; INPUT
PIN 23	A[18]	COMBINATORIAL; INPUT
PIN 13	/OE	COMBINATORIAL; INPUT
PIN 15	/PRG_REQ	COMBINATORIAL; OUTPUT
PIN 20	/BANK_PTR	REGISTERED; OUTPUT
PIN 19	PROGRAM	REGISTERED; OUTPUT
PIN 12	GND	
PIN 24	VCC	

;SPECIAL DEFINITIONS

STRING ACC_ST '/MSTATE[3] * MSTATE[2] * /MSTATE[1] * /MSTATE[0]'

STRING WC_ST '/MSTATE[3] * MSTATE[2] * /MSTATE[1] * MSTATE[0]'

STRING VALID_ADDR 'A[22] * A[21] * A[20] * A[19] * A[18]'

----- Boolean Equation Segment -----

EQUATIONS

PRG_REQ = VALID_ADDR

PROGRAM := PRG_REQ * /BINV * /REFRESH * PRG_DEC

BANK_PTR := ALE * /A[2] + /ALE * BANK_PTR * /(ACC_ST + WC_ST) + (ACC_ST + WC_ST) * /BANK_PTR

----- Simulation Segment -----

SIMULATION

"Memory Design for the 29K. This PAL will generate the signals that control the memory data and instruction buffers. It will also generate /RFCYC that indicates when the control logic is performing a memory refresh cycle."

;PALASM Design Description

----- Declaration Segment -----

```
TITLE CONTROL
PATTERN A
REVISION 1.0
COMPANY AMD INC.
DATE 07/12/91
CHIP _CONT29K PALCE16V8
```

----- PIN Declarations -----

```
PIN 1      CLOCKD      COMBINATORIAL ; INPUT
PIN 2      /MSTATE[3]  COMBINATORIAL ; INPUT
PIN 3      /MSTATE[2]  COMBINATORIAL ; INPUT
PIN 4      /MSTATE[1]  COMBINATORIAL ; INPUT
PIN 5      /MSTATE[0]  COMBINATORIAL ; INPUT
PIN 6      I_DL        COMBINATORIAL ; INPUT
PIN 7      LAR_WL      COMBINATORIAL ; INPUT
PIN 8      /REFRESH    COMBINATORIAL ; INPUT
PIN 9      /BANK_PTR   COMBINATORIAL ; INPUT
PIN 11     /RESET      COMBINATORIAL ; INPUT
PIN 12     /RFCYC      COMBINATORIAL ; OUTPUT
PIN 13     /OE_DIN     COMBINATORIAL ; OUTPUT
PIN 14     LE[0]       COMBINATORIAL ; OUTPUT
PIN 15     LE[1]       COMBINATORIAL ; OUTPUT
PIN 16     /OE_INS[0]  COMBINATORIAL ; OUTPUT
PIN 17     /OE_INS[1]  COMBINATORIAL ; OUTPUT
PIN 18     /OE_DOUT[0] COMBINATORIAL ; OUTPUT
PIN 19     /OE_DOUT[1] COMBINATORIAL ; OUTPUT
PIN 10     GND         COMBINATORIAL ; INPUT
PIN 20     VCC         COMBINATORIAL ; INPUT
```

;SPECIAL DEFINITIONS

```
STRING IDLE_ST '/MSTATE[3] * /MSTATE[2] * /MSTATE[1] * /MSTATE[0]'
STRING WS1_ST '/MSTATE[3] * /MSTATE[2] * MSTATE[1] * /MSTATE[0]'
STRING WS2_ST '/MSTATE[3] * MSTATE[2] * MSTATE[1] * /MSTATE[0]'
STRING ACC_ST '/MSTATE[3] * /MSTATE[2] * MSTATE[1] * /MSTATE[0]'
STRING PC1_ST '/MSTATE[3] * /MSTATE[2] * /MSTATE[1] * MSTATE[0]'
STRING PM_ST '/MSTATE[3] * /MSTATE[2] * /MSTATE[1] * /MSTATE[0]'
```

----- Boolean Equation Segment -----

EQUATIONS

```
OE_DOUT[0] = /RFCYC * /I_DL * LAR_WL * BANK_PTR * ACC_ST
OE_DOUT[1] = /RFCYC * /I_DL * LAR_WL * /BANK_PTR * ACC_ST
OE_INS[0] = /RFCYC * I_DL * BANK_PTR * ACC_ST
OE_INS[1] = /RFCYC * I_DL * /BANK_PTR * ACC_ST
OE_DIN = /RFCYC * /I_DL * /LAR_WL * (WS1_ST + WS2_ST + ACC_ST + PM_ST)
LE[0] = WS1_ST * BANK_PTR * /LAR_WL + ACC_ST * /BANK_PTR * /LAR_WL * CLOCKD
LE[1] = WS1_ST * /BANK_PTR * /LAR_WL + ACC_ST * BANK_PTR * /LAR_WL * CLOCKD
RFCYC = IDLE_ST * REFRESH * /RESET + RFCYC * /RESET * /PC1_ST
```

----- Simulation Segment -----

```
;SIMULATION
```



"29K memory design. This PAL decodes the addresses for programming request. It will also keep track of which bank is currently active through the output /BANK_PTR."

;PALASM Design Description

----- Declaration Segment -----

```
TITLE RASI
PATTERN A
REVISION 1.0
COMPANY AMD INC.
DATE 07/12/91
CHIP _RASI29K PAL16R4
```

----- PIN Declarations -----

```
PIN 1 SYCLK COMBINATORIAL ; INPUT
PIN 2 /MSTATE[3] COMBINATORIAL ; INPUT
PIN 3 /MSTATE[2] COMBINATORIAL ; INPUT
PIN 4 /MSTATE[1] COMBINATORIAL ; INPUT
PIN 5 /MSTATE[0] COMBINATORIAL ; INPUT
PIN 6 /BUSY COMBINATORIAL ; INPUT
PIN 7 /REFRESH COMBINATORIAL ; INPUT
PIN 8 /BINV COMBINATORIAL ; INPUT
PIN 9 /RESET COMBINATORIAL ; INPUT
PIN 11 /OE COMBINATORIAL ; INPUT
PIN 12 /CS COMBINATORIAL ; INPUT
PIN 13 /CH COMBINATORIAL ; INPUT
PIN 19 /IREQ COMBINATORIAL ; INPUT
PIN 17 /RASI REGISTERED ; OUTPUT
PIN 18 /RASI_OFF COMBINATORIAL ; OUTPUT
PIN 10 GND
PIN 20 VCC
```

;SPECIAL DEFINITIONS

```
STRING IDLE_ST '/MSTATE[3] * /MSTATE[2] * /MSTATE[1] * /MSTATE[0]'
STRING ACC_ST '/MSTATE[3] * MSTATE[2] * /MSTATE[1] * /MSTATE[0]'
STRING PC1_ST '/MSTATE[3] * /MSTATE[2] * /MSTATE[1] * MSTATE[0]'
STRING PM_ST 'MSTATE[3] * /MSTATE[2] * /MSTATE[1] * /MSTATE[0]'
STRING WC_ST '/MSTATE[3] * MSTATE[2] * /MSTATE[1] * MSTATE[0]'
```

----- Boolean Equation Segment -----

EQUATIONS

```
RASI := /(ACC_ST * REFRESH + ACC_ST * /BUSY * CS * /BINV * /CH + PM_ST * REFRESH + PM_ST * CS * /BINV * /CH *
/BUSY + PM_ST * CS * /BINV * /CH * BUSY * IREQ + IDLE_ST * /BUSY * /REFRESH * /(CS * /BINV) + RASI_OFF)
RASI_OFF = RESET + WC_ST + PC1_ST
```

----- Simulation Segment -----

SIMULATION

"32-bit Memory for the 29K. This PAL will generate the control signals back to the Am29000. It assumes that the control signals are externally combined to obtain the signal connected to the Am29000."

;PALASM Design Description

; Declaration Segment

TITLE CAS_ENABLE

PATTERN A

REVISION 1.0

COMPANY AMD INC.

DATE 07/12/91

CHIP _CAS_E29K PALCE20V8

; PIN Declarations

PIN 1	SYSCLK	COMBINATORIAL ; INPUT
PIN 2	CLOCKD	COMBINATORIAL ; INPUT
PIN 3	/MSTATE[2]	COMBINATORIAL ; INPUT
PIN 4	/MSTATE[1]	COMBINATORIAL ; INPUT
PIN 5	/MSTATE[0]	COMBINATORIAL ; INPUT
PIN 6	/BE[3]	COMBINATORIAL ; INPUT
PIN 7	/BE[2]	COMBINATORIAL ; INPUT
PIN 8	/BE[1]	COMBINATORIAL ; INPUT
PIN 9	/BE[0]	COMBINATORIAL ; INPUT
PIN 10	/CS	COMBINATORIAL ; INPUT
PIN 11	/RFCYC	COMBINATORIAL ; INPUT
PIN 13	/BINV	COMBINATORIAL ; INPUT
PIN 14	/RESET	COMBINATORIAL ; INPUT
PIN 23	/BANK_PTR	COMBINATORIAL ; INPUT
PIN 16	/CASEN1	COMBINATORIAL ; OUTPUT
PIN 17	/CASEN1[0]	COMBINATORIAL ; OUTPUT
PIN 18	/CASEN1[1]	COMBINATORIAL ; OUTPUT
PIN 19	/CASEN1[2]	COMBINATORIAL ; OUTPUT
PIN 20	/CASEN1[3]	COMBINATORIAL ; OUTPUT
PIN 21	CASEN0	COMBINATORIAL ; OUTPUT
PIN 15	MSEL	COMBINATORIAL ; OUTPUT
PIN 12	GND	; INPUT
PIN 24	VCC	; INPUT

;SPECIAL DEFINITIONS

STRING IDLE_ST '/MSTATE[2] * /MSTATE[1] * /MSTATE[0]'

STRING WS1_ST '/MSTATE[2] * MSTATE[1] * /MSTATE[0]'

STRING WS2_ST 'MSTATE[2] * MSTATE[1] * /MSTATE[0]'

STRING ACC_ST 'MSTATE[2] * /MSTATE[1] * /MSTATE[0]'

STRING PC1_ST '/MSTATE[2] * /MSTATE[1] * MSTATE[0]'

STRING PM_ST '/MSTATE[2] * /MSTATE[1] * /MSTATE[0]'

STRING WC_ST 'MSTATE[2] * /MSTATE[1] * MSTATE[0]'

; Boolean Equation Segment

EQUATIONS

MSEL = SYSCLK * WS1_ST + MSEL * /RESET * /PC1_ST

$$\text{CASEN0} = \text{WS2_ST} * \text{BANK_PTR} * / \text{SYSCLK} + \text{ACC_ST} * / \text{SYSCLK} + \text{CASEN0} * / (\text{SYSCLK} * \text{CASEN1} + \text{PC1_ST} + \text{PM_ST} * / \text{CS} * / \text{BINV} * / \text{CLOCKD} * / \text{SYSCLK})$$

$$\text{CASEN1} = / \text{SYSCLK} * \text{BANK_PTR} * (\text{CASEN1}[0] + \text{CASEN1}[1] + \text{CASEN1}[2] + \text{CASEN1}[3]) * \text{ACC_ST} + \text{PM_ST} * / \text{CASEN0} * / \text{SYSCLK} + \text{SYSCLK} * \text{CASEN1}$$

```

CASEN1[0] = /RESET * /RFCYC * BE[0] * WS2_ST * /BANK_PTR * /SYSCLK + /RESET * /RFCYC * BE[0] * ACC_ST * /SYSCLK +
/RESET * PM_ST * CASEN1[0] * /(CS * /BINV * /CLOCKD * /SYSCLK) + /RESET * CASEN1 * CASEN1[0] *
ACC_ST

CASEN1[1] = /RESET * /RFCYC * BE[1] * WS2_ST * /BANK_PTR * /SYSCLK + /RESET * /RFCYC * BE[1] * ACC_ST *
/RESET * PM_ST * CASEN1[1] * /(CS * /BINV * /CLOCKD * /SYSCLK) + /RESET * CASEN1 *
BE[1] * ACC_ST

CASEN1[2] = /RESET * /RFCYC * BE[2] * WS2_ST * /BANK_PTR * /SYSCLK + /RESET * /RFCYC * BE[2] * ACC_ST *
/RESET * PM_ST * CASEN1[2] * /(CS * /BINV * /CLOCKD * /SYSCLK) + /RESET * CASEN1 * BE[2]
* ACC_ST

CASEN1[3] = /RESET * /RFCYC * BE[3] * WS2_ST * /BANK_PTR * /SYSCLK + /RESET * /RFCYC * BE[3] * ACC_ST *
/RESET * PM_ST * CASEN1[3] * /(CS * /BINV * /CLOCKD * /SYSCLK) + /RESET * CASEN1 * BE[3]
* ACC_ST

```

Simulation Segment

SIMULATION

SIMULATION

Simulation Segment

Boolean Equation Segment

EQUATIONS

```

IRDY = /RFCYC * I_DL * ACC_ST
BACK = /RFCYC * I_DL * /IRREQ * (WS1_ST + WS2_ST) + BACK * /IRREQ * I_DL * /BUSY
DRDY = /RFCYC * A_DL * /LAR_WL * WS2_ST + /RFCYC * A_DL * /LAR_WL * ACC_ST
* BUSY + /RFCYC * A_DL * LAR_WL * ACC_ST
DBACK = /RFCYC * A_DL * /DBREQ * (WS1_ST + WS2_ST) + DBACK * /DBREQ * A_DL * /BUSY

```

"Memory System for the Am29000. This PAL decodes valid data and instruction memory requests. It will also partially decode programming requests."

;PALASM Design Description

Declaration Segment

TITLE READY

PATTERN A

REVISION 1.0

COMPANY AMD INC.

DATE 07/12/91

CHIP _READY29K PALCE16V8

PIN Declarations

PIN 1	I_DL	COMBINATORIAL ; INPUT
PIN 2	/IBREQ	COMBINATORIAL ; INPUT
PIN 3	/DBREQ	COMBINATORIAL ; INPUT
PIN 4	/MSTATE[2]	COMBINATORIAL ; INPUT
PIN 5	/MSTATE[1]	COMBINATORIAL ; INPUT
PIN 6	/MSTATE[0]	COMBINATORIAL ; INPUT
PIN 7	LAR_WL	COMBINATORIAL ; INPUT
PIN 8	/IREQ	COMBINATORIAL ; INPUT
PIN 9	/RFCYC	COMBINATORIAL ; INPUT
PIN 11	/BUSY	COMBINATORIAL ; INPUT
PIN 16	/IBACK	COMBINATORIAL ; OUTPUT
PIN 17	/DBACK	COMBINATORIAL ; OUTPUT
PIN 18	/IRDY	COMBINATORIAL ; OUTPUT
PIN 19	/DRDY	COMBINATORIAL ; OUTPUT
PIN 10	GND	; INPUT
PIN 20	VCC	; INPUT

;SPECIAL DEFINITIONS

STRING WS1_ST 'MSTATE[2] * MSTATE[1] * /MSTATE[0]'

STRING WS2_ST 'MSTATE[2] * MSTATE[1] * /MSTATE[0]'

STRING ACC_ST 'MSTATE[2] * /MSTATE[1] * /MSTATE[0]'

Boolean Equation Segment

EQUATIONS

IRDY = /RFCYC * I_DL * ACC_ST

IBACK = /RFCYC * I_DL * IBREQ * (WS1_ST + WS2_ST) + IBACK * /IREQ * I_DL * BUSY

DRDY = /RFCYC * /I_DL * /LAR_WL * WS2_ST + /RFCYC * /I_DL * /LAR_WL * ACC_ST
* BUSY + /RFCYC * /I_DL * LAR_WL * ACC_ST

DBACK = /RFCYC * /I_DL * DBREQ * (WS1_ST + WS2_ST) + DBACK * DBREQ * /I_DL * BUSY

Simulation Segment

SIMULATION



"Memory System for the Am29000. This PAL decodes valid data and instruction memory requests. It will also partially decode programing requests."

;PALASM Design Description

----- Declaration Segment -----

TITLE MEMORY_STATE_MACHINE

PATTERN A

REVISION 1.0

COMPANY AMD INC.

DATE 07/12/91

CHIP _STATE29K PALCE16V8

----- PIN Declarations -----

PIN 1	SYCLK	COMBINATORIAL ; INPUT
PIN 2	/CS	COMBINATORIAL ; INPUT
PIN 3	/DBREQ	COMBINATORIAL ; INPUT
PIN 4	LAR_WL	COMBINATORIAL ; INPUT
PIN 5	/IBREQ	COMBINATORIAL ; INPUT
PIN 6	/CH	COMBINATORIAL ; INPUT
PIN 7	/IREQ	COMBINATORIAL ; INPUT
PIN 8	/RESET	COMBINATORIAL ; INPUT
PIN 9	/REFRESH	COMBINATORIAL ; INPUT
PIN 11	/OE	COMBINATORIAL ; INPUT
PIN 19	/BINV	COMBINATORIAL ; INPUT
PIN 12	/NOT_BUSY	COMBINATORIAL ; OUTPUT
PIN 13	/BUSY	REGISTERED ; OUTPUT
PIN 14	/L_DL	REGISTERED ; OUTPUT
PIN 15	/MSTATE[3]	REGISTERED ; OUTPUT
PIN 16	/MSTATE[2]	REGISTERED ; OUTPUT
PIN 17	/MSTATE[1]	REGISTERED ; OUTPUT
PIN 18	/MSTATE[0]	REGISTERED ; OUTPUT
PIN 10	GND	; INPUT
PIN 20	VCC	; INPUT

;SPECIAL DEFINITIONS

```
STRING IDLE_ST 'MSTATE[3] * MSTATE[2] * MSTATE[1] * MSTATE[0]'
STRING WS1_ST 'MSTATE[3] * MSTATE[2] * MSTATE[1] * MSTATE[0]'
STRING WS2_ST 'MSTATE[3] * MSTATE[2] * MSTATE[1] * MSTATE[0]'
STRING ACC_ST 'MSTATE[3] * MSTATE[2] * MSTATE[1] * MSTATE[0]'
STRING PC1_ST 'MSTATE[3] * MSTATE[2] * MSTATE[1] * MSTATE[0]'
STRING PM_ST 'MSTATE[3] * MSTATE[2] * MSTATE[1] * MSTATE[0]'
STRING WC_ST 'MSTATE[3] * MSTATE[2] * MSTATE[1] * MSTATE[0]'
STRING IDLE '#B0000'
STRING WS1 '#B0010'
STRING WS2 '#B0110'
STRING ACC '#B0100'
STRING PC1 '#B0001'
STRING WC '#B0101'
STRING PM '#B1000'
```

----- Boolean Equation Segment -----

EQUATIONS

BUSY := /BUSY * (IDLE_ST + PM_ST + ACC_ST) * CS * /BINV * /RESET + BUSY * /NOT_BUSY

```

NOT_BUSY = RESET + WS2_ST * I_DL * /LAR_WL * /DBREQ + ACC_ST * I_DL * /DBREQ + ACC_ST * /I_DL * CS * /BINV +
    PM_ST * /I_DL * CS * /BINV + PM_ST * IREQ * /CS
I_DL := (IDLE_ST + PM_ST + ACC_ST) * /BUSY * /IREQ * CS * /BINV + PM_ST * BUSY * IREQ * /CS + (IDLE_ST +
    ACC_ST) * /CS * /BUSY + I_DL * (WS1_ST + WS2_ST + ACC_ST * BUSY + PC1_ST * BUSY + IDLE_ST * BUSY) +
    RESET
CASE (MSTATE[3..0])
BEGIN
IDLE:
BEGIN
    IF (RESET * (REFRESH + CS * /BINV + BUSY)) THEN
        BEGIN
            MSTATE[3..0] := WS1
        END
        IF (/REFRESH * /BUSY * (/CS + BINV)) THEN
            BEGIN
                MSTATE[3..0] := IDLE
            END
        END
WS1:
BEGIN
    IF (RESET) THEN
        BEGIN
            MSTATE[3..0] := IDLE
        END
    ELSE
        BEGIN
            MSTATE[3..0] := WS2
        END
    END
WS2:
BEGIN
    IF (RESET) THEN
        BEGIN
            MSTATE[3..0] := IDLE
        END
    ELSE
        BEGIN
            MSTATE[3..0] := ACC
        END
    END
ACC:
BEGIN
    IF (RESET) THEN
        BEGIN
            MSTATE[3..0] := IDLE
        END
    IF (/RESET * REFRESH) THEN
        BEGIN
            IF (I_DL * /LAR_WL * BUSY) THEN

```



```

    BEGIN
        MSTATE[3..0] := WC
    END
ELSE
    BEGIN
        MSTATE[3..0] := PC1
    END
END
IF (/RESET * /REFRESH * BUSY * (/I_DL * IBREQ + I_DL * DBREQ)) THEN
    BEGIN
        MSTATE[3..0] := ACC
    END
IF (/RESET * /REFRESH * (/I_DL * /IBREQ * BUSY + I_DL * /DBREQ)) THEN
    BEGIN
        MSTATE[3..0] := PM
    END
IF (/RESET * /REFRESH * /BUSY * CH * CS * /BINV) THEN
    BEGIN
        MSTATE[3..0] := WS2
    END
IF (/RESET * /REFRESH * /BUSY * /CH * CS * /BINV) THEN
    BEGIN
        MSTATE[3..0] := PC1
    END
IF (/RESET * /REFRESH * /BUSY * /(CS * /BINV)) THEN
    BEGIN
        MSTATE[3..0] := PM
    END
END
PC1:
BEGIN
    MSTATE[3..0] := IDLE
END
PM:
BEGIN
    IF (/RESET * (REFRESH + (/BUSY + IREQ) * CS * /CH * /BINV)) THEN
        BEGIN
            MSTATE[3..0] := PC1
        END
    IF (/RESET * /REFRESH * (/BUSY + IREQ) * CS * CH * /BINV) THEN
        BEGIN
            MSTATE[3..0] := WS2
        END
    IF (/RESET * /REFRESH * BUSY * IBREQ * /I_DL) THEN
        BEGIN
            MSTATE[3..0] := ACC
        END
    IF (/RESET * /REFRESH * /(CS * /BINV * (/BUSY + IREQ) * (/IBREQ * /I_DL * BUSY))) THEN
        BEGIN
            MSTATE[3..0] := PM
        END

```




"Memory System for the Am29000. This PAL decodes valid data and instruction memory requests. It will also partially decode programming requests."

;PALASM Design Description

----- Declaration Segment -----

TITLE TIMER

PATTERN A

REVISION 1.0

COMPANY AMD INC.

DATE 07/12/91

CHIP _TIMER29K PAL20X10

----- PIN Declarations -----

PIN	Signal	Function
PIN 1	SYSCLK	COMBINATORIAL ; INPUT
PIN 2	/RESET	COMBINATORIAL ; INPUT
PIN 3	/RFCYC	COMBINATORIAL ; INPUT
PIN 4	/MSTATE[3]	COMBINATORIAL ; INPUT
PIN 5	/MSTATE[2]	COMBINATORIAL ; INPUT
PIN 6	/MSTATE[1]	COMBINATORIAL ; INPUT
PIN 7	/MSTATE[0]	COMBINATORIAL ; INPUT
PIN 13	/OE	COMBINATORIAL ; INPUT
PIN 14	/REFRESH	REGISTERED ; OUTPUT
PIN 15	/RCT[0]	REGISTERED ; OUTPUT
PIN 16	/RCT[1]	REGISTERED ; OUTPUT
PIN 17	/RCT[2]	REGISTERED ; OUTPUT
PIN 18	/RCT[3]	REGISTERED ; OUTPUT
PIN 19	/RCT[4]	REGISTERED ; OUTPUT
PIN 20	/RCT[5]	REGISTERED ; OUTPUT
PIN 21	/RCT[6]	REGISTERED ; OUTPUT
PIN 22	/RCT[7]	REGISTERED ; OUTPUT
PIN 12	GND	COMBINATORIAL ; INPUT
PIN 24	VCC	COMBINATORIAL ; INPUT

;SPECIAL DEFINITIONS

STRING ACC_ST '/MSTATE[3] * /MSTATE[2] * /MSTATE[1] * /MSTATE[0]

STRING START_REFRESH '/RCT[7] * /RCT[6] * /RCT[5] * /RCT[4] * /RCT[3] * /RCT[2] * /RCT[1] * /RCT[0]

STRING INIT '/RCT[7] * /RCT[6] * /RCT[5] * /RCT[4] * /RCT[3] * /RCT[2] * /RCT[1] * /RCT[0]

STRING ONE 'VCC'

STRING ZERO 'GND'

----- Boolean Equation Segment -----

EQUATIONS

REFRESH := (START_REFRESH * /RESET + REFRESH * /RESET) :+ (RFCYC * ACC_ST * /RESET)

RCT[0] := (/RCT[0] + INIT * ONE) :+ (INIT * ZERO)

RCT[1] := (/RCT[0] + INIT * ONE) :+ (RCT[1] + INIT * ZERO)

RCT[2] := (/RCT[1] * /RCT[0] + INIT * ZERO) :+ (RCT[2] + INIT * ONE)

RCT[3] := (/RCT[2] * /RCT[1] * /RCT[0] + INIT * ZERO) :+ (RCT[3] + INIT * ONE)

RCT[4] := (/RCT[3] * /RCT[2] * /RCT[1] * /RCT[0] + INIT * ZERO) :+ (RCT[4] + INIT * ONE)

RCT[5] := (/RCT[4] * /RCT[3] * /RCT[2] * /RCT[1] * /RCT[0] + INIT * ZERO) :+ (RCT[5] + INIT * ONE)

RCT[6] := (/RCT[5] * /RCT[4] * /RCT[3] * /RCT[2] * /RCT[1] * /RCT[0] + INIT * ONE) :+ (RCT[6] + INIT * ZERO)

RCT[7] := (/RCT[6] * /RCT[5] * /RCT[4] * /RCT[3] * /RCT[2] * /RCT[1] * /RCT[0] + INIT * ONE) :+ (RCT[7] + INIT * ZERO)

----- Simulation Segment -----

SIMULATION

Memory System for the Am29000. This PAL decodes valid data and instruction memory requests. It will also partially decode programming requests."

PALASM Design Description

```

;----- Declaration Segment -----
TITLE BYTE_ENABLE
PATTERN A
REVISION 1.0
COMPANY AMD INC.
DATE 07/12/91
CHIP _B_EN29K PALCE20V8

;----- PIN Declarations -----
PIN 1 ALE COMBINATORIAL ; INPUT
PIN 2 OPT[2] COMBINATORIAL ; INPUT
PIN 3 OPT[1] COMBINATORIAL ; INPUT
PIN 4 OPT[0] COMBINATORIAL ; INPUT
PIN 5 A[1] COMBINATORIAL ; INPUT
PIN 6 A[0] COMBINATORIAL ; INPUT
PIN 7 /CASN1[3] COMBINATORIAL ; INPUT
PIN 8 /CASN1[2] COMBINATORIAL ; INPUT
PIN 9 /CASN1[1] COMBINATORIAL ; INPUT
PIN 10 /CASN1[0] COMBINATORIAL ; INPUT
PIN 11 I_DL COMBINATORIAL ; INPUT
PIN 13 R_WL COMBINATORIAL ; INPUT
PIN 14 PROGRAM COMBINATORIAL ; INPUT
PIN 16 /RESET COMBINATORIAL ; INPUT
PIN 23 SYSCLK COMBINATORIAL ; INPUT
PIN 15 RL_CC COMBINATORIAL ; OUTPUT
PIN 17 LAR_WL COMBINATORIAL ; OUTPUT
PIN 21 /BE[3] COMBINATORIAL ; OUTPUT
PIN 20 /BE[2] COMBINATORIAL ; OUTPUT
PIN 19 /BE[1] COMBINATORIAL ; OUTPUT
PIN 18 /BE[0] COMBINATORIAL ; OUTPUT
PIN 12 GND ; INPUT
PIN 24 VCC ; INPUT

;SPECIAL DEFINITIONS
STRING ZERO 'GND'
STRING ONE 'VCC'

```

```

;----- Boolean Equation Segment -----
EQUATIONS
REFRESH = (START_REFRESH * RESET + REFRESH * REFRESH + (RCYC * ACC_ST * RESET)
RC[0] = (RC[0] + INT * ONE) < (INT * ZERO)
RC[1] = (RC[1] + INT * ONE) < (RC[1] + INT * ZERO)
RC[2] = (RC[2] + INT * ONE) < (RC[2] + INT * ZERO)
RC[3] = (RC[3] + INT * ONE) < (RC[3] + INT * ZERO)
RC[4] = (RC[4] + INT * ONE) < (RC[4] + INT * ZERO)
RC[5] = (RC[5] + INT * ONE) < (RC[5] + INT * ZERO)
RC[6] = (RC[6] + INT * ONE) < (RC[6] + INT * ZERO)
RC[7] = (RC[7] + INT * ONE) < (RC[7] + INT * ZERO)
RC[8] = (RC[8] + INT * ONE) < (RC[8] + INT * ZERO)
RC[9] = (RC[9] + INT * ONE) < (RC[9] + INT * ZERO)
RC[10] = (RC[10] + INT * ONE) < (RC[10] + INT * ZERO)
RC[11] = (RC[11] + INT * ONE) < (RC[11] + INT * ZERO)
RC[12] = (RC[12] + INT * ONE) < (RC[12] + INT * ZERO)
RC[13] = (RC[13] + INT * ONE) < (RC[13] + INT * ZERO)
RC[14] = (RC[14] + INT * ONE) < (RC[14] + INT * ZERO)
RC[15] = (RC[15] + INT * ONE) < (RC[15] + INT * ZERO)
RC[16] = (RC[16] + INT * ONE) < (RC[16] + INT * ZERO)
RC[17] = (RC[17] + INT * ONE) < (RC[17] + INT * ZERO)
RC[18] = (RC[18] + INT * ONE) < (RC[18] + INT * ZERO)
RC[19] = (RC[19] + INT * ONE) < (RC[19] + INT * ZERO)
RC[20] = (RC[20] + INT * ONE) < (RC[20] + INT * ZERO)
RC[21] = (RC[21] + INT * ONE) < (RC[21] + INT * ZERO)
RC[22] = (RC[22] + INT * ONE) < (RC[22] + INT * ZERO)
RC[23] = (RC[23] + INT * ONE) < (RC[23] + INT * ZERO)
RC[24] = (RC[24] + INT * ONE) < (RC[24] + INT * ZERO)
RC[25] = (RC[25] + INT * ONE) < (RC[25] + INT * ZERO)
RC[26] = (RC[26] + INT * ONE) < (RC[26] + INT * ZERO)
RC[27] = (RC[27] + INT * ONE) < (RC[27] + INT * ZERO)
RC[28] = (RC[28] + INT * ONE) < (RC[28] + INT * ZERO)
RC[29] = (RC[29] + INT * ONE) < (RC[29] + INT * ZERO)
RC[30] = (RC[30] + INT * ONE) < (RC[30] + INT * ZERO)
RC[31] = (RC[31] + INT * ONE) < (RC[31] + INT * ZERO)
RC[32] = (RC[32] + INT * ONE) < (RC[32] + INT * ZERO)
RC[33] = (RC[33] + INT * ONE) < (RC[33] + INT * ZERO)
RC[34] = (RC[34] + INT * ONE) < (RC[34] + INT * ZERO)
RC[35] = (RC[35] + INT * ONE) < (RC[35] + INT * ZERO)
RC[36] = (RC[36] + INT * ONE) < (RC[36] + INT * ZERO)
RC[37] = (RC[37] + INT * ONE) < (RC[37] + INT * ZERO)
RC[38] = (RC[38] + INT * ONE) < (RC[38] + INT * ZERO)
RC[39] = (RC[39] + INT * ONE) < (RC[39] + INT * ZERO)
RC[40] = (RC[40] + INT * ONE) < (RC[40] + INT * ZERO)
RC[41] = (RC[41] + INT * ONE) < (RC[41] + INT * ZERO)
RC[42] = (RC[42] + INT * ONE) < (RC[42] + INT * ZERO)
RC[43] = (RC[43] + INT * ONE) < (RC[43] + INT * ZERO)
RC[44] = (RC[44] + INT * ONE) < (RC[44] + INT * ZERO)
RC[45] = (RC[45] + INT * ONE) < (RC[45] + INT * ZERO)
RC[46] = (RC[46] + INT * ONE) < (RC[46] + INT * ZERO)
RC[47] = (RC[47] + INT * ONE) < (RC[47] + INT * ZERO)
RC[48] = (RC[48] + INT * ONE) < (RC[48] + INT * ZERO)
RC[49] = (RC[49] + INT * ONE) < (RC[49] + INT * ZERO)
RC[50] = (RC[50] + INT * ONE) < (RC[50] + INT * ZERO)
RC[51] = (RC[51] + INT * ONE) < (RC[51] + INT * ZERO)
RC[52] = (RC[52] + INT * ONE) < (RC[52] + INT * ZERO)
RC[53] = (RC[53] + INT * ONE) < (RC[53] + INT * ZERO)
RC[54] = (RC[54] + INT * ONE) < (RC[54] + INT * ZERO)
RC[55] = (RC[55] + INT * ONE) < (RC[55] + INT * ZERO)
RC[56] = (RC[56] + INT * ONE) < (RC[56] + INT * ZERO)
RC[57] = (RC[57] + INT * ONE) < (RC[57] + INT * ZERO)
RC[58] = (RC[58] + INT * ONE) < (RC[58] + INT * ZERO)
RC[59] = (RC[59] + INT * ONE) < (RC[59] + INT * ZERO)
RC[60] = (RC[60] + INT * ONE) < (RC[60] + INT * ZERO)
RC[61] = (RC[61] + INT * ONE) < (RC[61] + INT * ZERO)
RC[62] = (RC[62] + INT * ONE) < (RC[62] + INT * ZERO)
RC[63] = (RC[63] + INT * ONE) < (RC[63] + INT * ZERO)
RC[64] = (RC[64] + INT * ONE) < (RC[64] + INT * ZERO)
RC[65] = (RC[65] + INT * ONE) < (RC[65] + INT * ZERO)
RC[66] = (RC[66] + INT * ONE) < (RC[66] + INT * ZERO)
RC[67] = (RC[67] + INT * ONE) < (RC[67] + INT * ZERO)
RC[68] = (RC[68] + INT * ONE) < (RC[68] + INT * ZERO)
RC[69] = (RC[69] + INT * ONE) < (RC[69] + INT * ZERO)
RC[70] = (RC[70] + INT * ONE) < (RC[70] + INT * ZERO)
RC[71] = (RC[71] + INT * ONE) < (RC[71] + INT * ZERO)
RC[72] = (RC[72] + INT * ONE) < (RC[72] + INT * ZERO)
RC[73] = (RC[73] + INT * ONE) < (RC[73] + INT * ZERO)
RC[74] = (RC[74] + INT * ONE) < (RC[74] + INT * ZERO)
RC[75] = (RC[75] + INT * ONE) < (RC[75] + INT * ZERO)
RC[76] = (RC[76] + INT * ONE) < (RC[76] + INT * ZERO)
RC[77] = (RC[77] + INT * ONE) < (RC[77] + INT * ZERO)
RC[78] = (RC[78] + INT * ONE) < (RC[78] + INT * ZERO)
RC[79] = (RC[79] + INT * ONE) < (RC[79] + INT * ZERO)
RC[80] = (RC[80] + INT * ONE) < (RC[80] + INT * ZERO)
RC[81] = (RC[81] + INT * ONE) < (RC[81] + INT * ZERO)
RC[82] = (RC[82] + INT * ONE) < (RC[82] + INT * ZERO)
RC[83] = (RC[83] + INT * ONE) < (RC[83] + INT * ZERO)
RC[84] = (RC[84] + INT * ONE) < (RC[84] + INT * ZERO)
RC[85] = (RC[85] + INT * ONE) < (RC[85] + INT * ZERO)
RC[86] = (RC[86] + INT * ONE) < (RC[86] + INT * ZERO)
RC[87] = (RC[87] + INT * ONE) < (RC[87] + INT * ZERO)
RC[88] = (RC[88] + INT * ONE) < (RC[88] + INT * ZERO)
RC[89] = (RC[89] + INT * ONE) < (RC[89] + INT * ZERO)
RC[90] = (RC[90] + INT * ONE) < (RC[90] + INT * ZERO)
RC[91] = (RC[91] + INT * ONE) < (RC[91] + INT * ZERO)
RC[92] = (RC[92] + INT * ONE) < (RC[92] + INT * ZERO)
RC[93] = (RC[93] + INT * ONE) < (RC[93] + INT * ZERO)
RC[94] = (RC[94] + INT * ONE) < (RC[94] + INT * ZERO)
RC[95] = (RC[95] + INT * ONE) < (RC[95] + INT * ZERO)
RC[96] = (RC[96] + INT * ONE) < (RC[96] + INT * ZERO)
RC[97] = (RC[97] + INT * ONE) < (RC[97] + INT * ZERO)
RC[98] = (RC[98] + INT * ONE) < (RC[98] + INT * ZERO)
RC[99] = (RC[99] + INT * ONE) < (RC[99] + INT * ZERO)

```

EQUATIONS

$$\text{BE}[0] = / \text{RESET} * \text{ALE} * (\text{R_WL} + / \text{OPT}[2] * / \text{OPT}[1] * / \text{OPT}[0] + / \text{OPT}[2] * / \text{OPT}[1] * \text{OPT}[0] * \text{A}[1] * \text{A}[0] * \text{ZERO} + / \text{OPT}[2] * \text{OPT}[1] * / \text{OPT}[0] * \text{A}[1] * \text{ZERO} + / \text{OPT}[2] * / \text{OPT}[1] * \text{OPT}[0] * \text{A}[1] * \text{A}[0] * \text{ONE} + / \text{OPT}[2] * \text{OPT}[1] * / \text{OPT}[0] * \text{A}[1] * \text{ONE}) + \text{BE}[0] * / \text{ALE}$$

$$\text{BE}[1] = / \text{RESET} * \text{ALE} * (\text{R_WL} + / \text{OPT}[2] * / \text{OPT}[1] * / \text{OPT}[0] + / \text{OPT}[2] * / \text{OPT}[1] * \text{OPT}[0] * \text{A}[1] * \text{A}[0] * \text{ZERO} + / \text{OPT}[2] * \text{OPT}[1] * / \text{OPT}[0] * \text{A}[1] * \text{ZERO} + / \text{OPT}[2] * / \text{OPT}[1] * \text{OPT}[0] * \text{A}[1] * \text{A}[0] * \text{ONE} + / \text{OPT}[2] * \text{OPT}[1] * / \text{OPT}[0] * \text{A}[1] * \text{ONE}) + \text{BE}[1] * / \text{ALE}$$

$$\text{BE}[2] = / \text{RESET} * \text{ALE} * (\text{R_WL} + / \text{OPT}[2] * / \text{OPT}[1] * / \text{OPT}[0] + / \text{OPT}[2] * / \text{OPT}[1] * \text{OPT}[0] * \text{A}[1] * \text{A}[0] * \text{ZERO} + / \text{OPT}[2] * \text{OPT}[1] * / \text{OPT}[0] * \text{A}[1] * \text{ZERO} + / \text{OPT}[2] * / \text{OPT}[1] * \text{OPT}[0] * \text{A}[1] * \text{A}[0] * \text{ONE} + / \text{OPT}[2] * \text{OPT}[1] * / \text{OPT}[0] * \text{A}[1] * \text{ONE}) + \text{BE}[2] * / \text{ALE}$$

$$\text{BE}[3] = / \text{RESET} * \text{ALE} * (\text{R_WL} + / \text{OPT}[2] * / \text{OPT}[1] * / \text{OPT}[0] + / \text{OPT}[2] * / \text{OPT}[1] * \text{OPT}[0] * \text{A}[1] * \text{A}[0] * \text{ZERO} + / \text{OPT}[2] * \text{OPT}[1] * / \text{OPT}[0] * \text{A}[1] * \text{ZERO} + / \text{OPT}[2] * / \text{OPT}[1] * \text{OPT}[0] * \text{A}[1] * \text{A}[0] * \text{ONE} + / \text{OPT}[2] * \text{OPT}[1] * / \text{OPT}[0] * \text{A}[1] * \text{ONE}) + \text{BE}[3] * / \text{ALE}$$

$$\text{RL_CC} = / \text{PROGRAM} * (/ \text{CASN1}[0] + \text{CASN1}[1] + \text{CASN1}[2] + \text{CASN1}[3]) + \text{PROGRAM} * / \text{SYSCLK}$$

$$\text{LAR_WL} = \text{ALE} * (/ \text{I_DL} + / \text{I_DL} * \text{R_WL}) + \text{LAR_WL} * / \text{ALE}$$

; _____ Simulation Segment _____

SIMULATION

; _____

Am29C668 Configurable Dynamic Memory Controller to 80C286 Microprocessor Interface



INTRODUCTION

The interface between the Am29C668 4-MBit Configurable Dynamic Memory Controller/Driver (CDMC) and the AMD 80C286 microprocessor was designed to provide maximum performance at reasonable cost. This design is as general as possible so that the user may tailor his implementation to a specific memory system. Possible changes to the design are discussed with associated system requirements and implications. A block diagram, timing analyses and logic equations necessary to implement the design are included. This design requires a minimum number of external devices to perform the interface and glue functions: two PAL™ devices (one PAL16R8, one PAL20R4), a 555 timer and an inverter.

Distinctive Characteristics

- Am29C668 4-MBit Configurable Dynamic Memory Controller/Driver, With Selectable Auto-Timing or External-Timing Mode
- 20-MHz 80C286 Microprocessor (may be upgraded to 25 MHz)
- 85-ns Fast-Page-Mode 1 Mbit x 1 DRAM
- One Wait-State Initial Accesses With Zero Wait State Subsequent Page-Mode Accesses
- 8-Mbyte Dynamic Memory per Am29C668

MEMORY ARCHITECTURE OVERVIEW

To obtain the maximum memory throughput but still maintain a reasonable cost, 85-ns fast page-mode 1-MBit DRAMs are used. The 80C286 requires a minimum of two processor cycles per access. If additional cycles are needed, the external logic holds **READY** inactive and the processor inserts wait states until **READY** is asserted. For this memory design, the 20-MHz 80C286 completes the initial access to memory in one wait state (three processor cycles total). The subsequent accesses within the page are performed with no wait states (two processor cycles).

The page size for a 1-Mbit DRAM is 1024 bits or 1 Kbits. This memory is 16 bits wide, therefore the page size is 2 Kbytes. The Am29C668 detects accesses within the same page via on-chip cache-mode operation; consequently, page-mode DRAMs appear to the processor as

if they are fast cache memories. When a new address is latched, it is compared with the previous row and bank address; if the addresses are to the same row and bank, the Cache Hit signal **CH** is asserted. The memory state machine immediately begins the next access. An access outside the page, a page miss, causes the memory controller to perform the **RAS** precharge for the DRAMs. The total access time on a page miss requires five processor-clock cycles, one for decoding, two cycles for the **RAS** precharge and two for the data access. This method of accessing memory results in shorter access times than memories using normal DRAM accesses. For certain systems, this memory can result in near-zero wait-state accesses. Only static RAMs can guarantee zero wait-state accesses. The actual performance of the memory depends upon the instruction mix of the programs executed.

The memory array consists of four banks. Each bank contains 2 Mbytes or 1 Mword (16-bits) of memory that provides a maximum memory size of 8 Mbytes or 4 Mwords per Am29C668 controller.

A 20-MHz system was selected since high performance is achieved with 85-ns DRAMs. For a 25-MHz processor using the same memory architecture, a single-wait-state memory would require 60-ns DRAMs. The State PAL would have to be implemented using a PAL1 6L8-7 device to meet the faster clock rate and shorter set-up times. This would be very expensive. For such a system, a cache could be a more cost-effective solution than using fast expensive DRAMs.

The 80C286 generates its internal processor clock from an external oscillator. The external clock oscillates at twice the speed of the 80C286 processor clock. The external crystal also provides the clock for the memory control logic. The oscillator signal is inverted to obtain **MCLOCK** for the memory control logic. The 80C286 20-MHz clock is referred to as the processor clock or **CLK** and the memory controller 40-MHz clock is called the memory clock or **MCLOCK** to avoid confusion.

FUNCTIONAL DESCRIPTION

The primary data paths and functional elements are shown in Figure 1. The following discussion describes each subsection of the block diagram, including the control logic, timer, buffers and memory array.

03552-001A

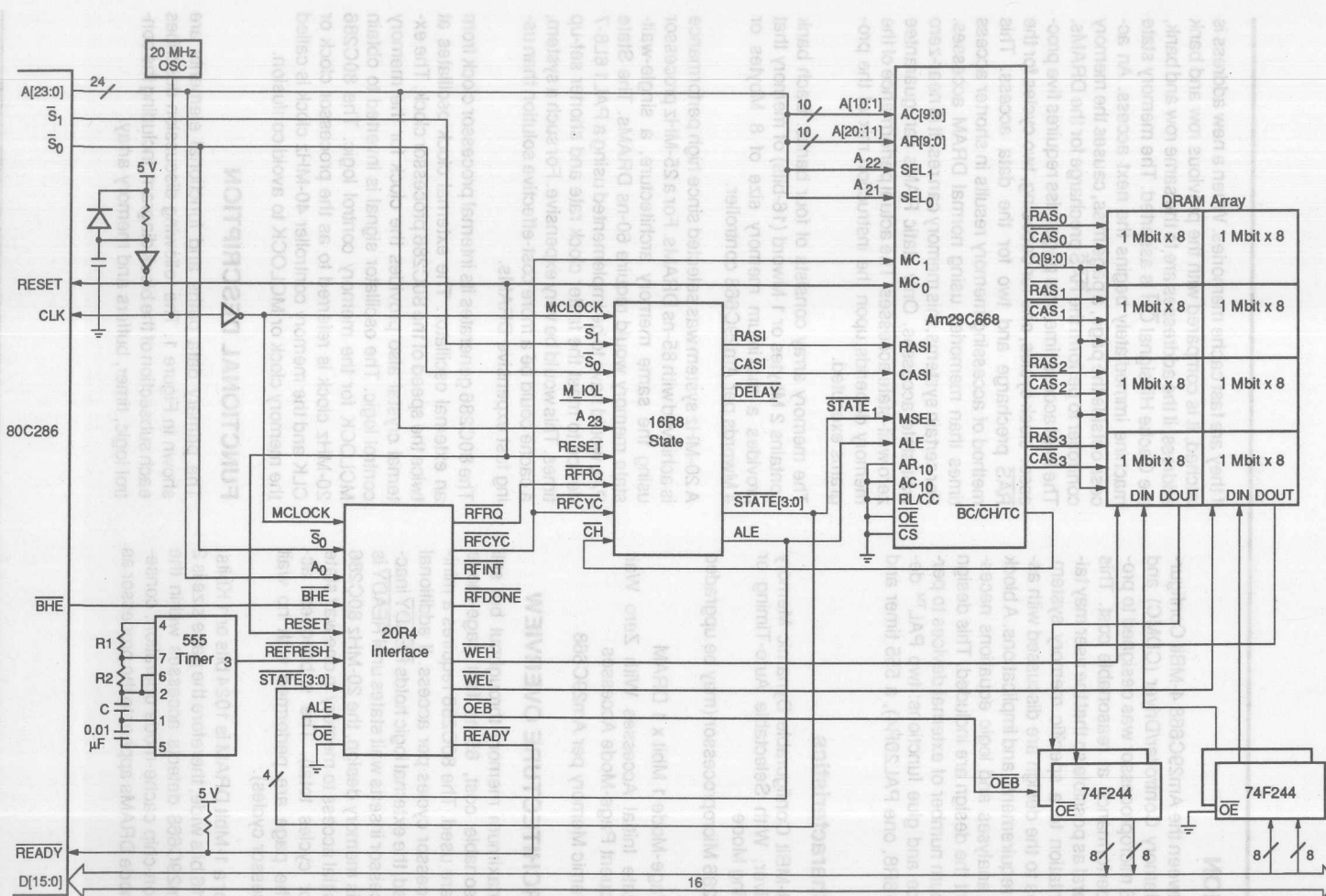


Figure 1. Interface Block Diagram

Am29C668 CDMC

The Am29C668 CDMC generates the $\overline{\text{RAS}}$, $\overline{\text{CAS}}$ and address signals to the DRAM array; no external drivers are needed. Additionally, the Am29C668 generates the row addresses during $\overline{\text{RAS}}$ -only refreshes from its internal row-refresh counter. The Am29C668 timing is controlled externally by the State PAL and the Interface PAL.

The State PAL generates the RASI, CASI, MSEL and ALE inputs to the Am29C668. RASI controls the $\overline{\text{RAS}}_n$ outputs. When RASI is asserted, the proper $\overline{\text{RAS}}_n$ outputs are enabled. CASI similarly controls the $\overline{\text{CAS}}_n$ outputs. During refresh and reset, the $\overline{\text{CAS}}_n$ outputs are always suppressed by the Am29C668. MSEL controls the multiplexing of the row and column address. When MSEL is Low, the row address is output on the Q[9:0] outputs of the Am29C668; when MSEL is High, the column address is output. If the Am29C668 is in the read/write mode, the address is taken from the address latch; in the refresh mode, the address is taken from the row-refresh counter.

The Interface PAL controls the Am29C668 mode of operation: read/write, refresh or reset. The Refresh Cycle output RFCYC is connected to the MC₁ input of the Am29C668. When $\overline{\text{RFCYC}}$ is active Low, the Am29C668 performs refresh cycles. When $\overline{\text{RFCYC}}$ is not active, i.e., High, the Am29C668 is either in read/write mode or reset. If RESET is asserted, the control logic forces the Am29C668 into the reset mode, resetting the refresh counters to zero and reconfiguring the CDMC to the default mode. The memory controller initiates "wake up" cycles to the DRAMs until RESET is deasserted. If RESET and $\overline{\text{RFCYC}}$ are deasserted, the Am29C668 is in the read/write mode.

The $\overline{\text{CH}}$ signal from the Am29C668 is used to determine if the current address in the input latch has the same row and bank address as the previous memory access. If the current access is to the same bank and row, $\overline{\text{CH}}$ is asserted and a page-mode access is initiated. If $\overline{\text{CH}}$ is deasserted, $\overline{\text{RAS}}$ must be precharged and a normal access occurs.

For this application, the Am29C668 is used in the default configuration, with external timing, 4-bank configuration, 1-Mbit DRAMs, $\overline{\text{CAS}}$ bank decoding, $\overline{\text{RAS}}$ -only refresh and cache mode. It is possible to reconfigure the Am29C668 to provide additional features not used in this application (see the Am29C668 data sheet).

Interface PAL

This PAL generates the ready signal to the 80C286, refresh signals to the control logic, the output enable to the buffers, and write enables to the DRAMs. The following inputs are used:

MCLOCK Inverted 40-MHz system clock

$\overline{\text{S0}}$	Status, from 80C286
A[0]	Address bit 0, from 80C286
$\overline{\text{BHE}}$	Bus High Enable, from 80C286
RESET	System Reset
$\overline{\text{REFRESH}}$	Refresh, from 555 timer
STATE[3:0]	State variables, from State PAL
ALE	Address Latch Enable, from State PAL
The following outputs are generated:	
$\overline{\text{READY}}$	Ready, to 80C286
$\overline{\text{RFINT}}$	Refresh Intermediate, used to synchronize refresh requests
$\overline{\text{RFRQ}}$	Refresh Request, to State PAL
$\overline{\text{RFCYC}}$	Refresh Cycle, Indicates if access is a refresh cycle
$\overline{\text{RFDONE}}$	Refresh Done
$\overline{\text{WEL}}$	Write Enable Low, to DRAM array
$\overline{\text{WEH}}$	Write Enable High, to DRAM array
$\overline{\text{OEB}}$	Output Enable Buffer, to Data Output Buffers

The $\overline{\text{READY}}$ output is a three-state output and requires an external pull-up resistor to keep it deasserted. When $\overline{\text{READY}}$ is asserted, it signals the 80C286 that the current memory access is completed.

$\overline{\text{OEB}}$ enables the outputs of the data drivers during processor reads. The memory has separate data input and output lines since all write cycles are late write cycles.

$\overline{\text{WEL}}$ and $\overline{\text{WEH}}$ control the write cycles to the memory data bytes; $\overline{\text{WEL}}$ controls the lower memory byte and $\overline{\text{WEH}}$ controls the upper byte. $\overline{\text{WEL}}$ is asserted during write cycles when A[0] is Low; $\overline{\text{WEH}}$ is asserted during write cycles when $\overline{\text{BHE}}$ is Low. Since the status bit must be decoded to determine if the access is a read or write cycle, $\overline{\text{CAS}}_n$ to the DRAMs may occur before $\overline{\text{WEL}}$ or $\overline{\text{WEH}}$ is asserted, enabling the DRAMs output drivers. This requires that the data input and data output lines be separate.

$\overline{\text{RFINT}}$, $\overline{\text{RFRQ}}$, $\overline{\text{RFCYC}}$ and $\overline{\text{RFDONE}}$ control the refresh cycles. The 555 timer controls the refresh interval and generates refresh requests by driving the $\overline{\text{REFRESH}}$ input Low. This signal is asynchronous to the memory clock, therefore it must be synchronized. Two internal D-flip-flops, $\overline{\text{RFINT}}$ and $\overline{\text{RFRQ}}$, are used to synchronize RASI the refresh requests to avoid any metastability. $\overline{\text{RFDONE}}$ is used to prevent $\overline{\text{REFRESH}}$, which is active for a long time, from generating multiple refresh cycles. Once the CASI refresh cycle is initiated, $\overline{\text{RFDONE}}$ is as-

serted and remains so until $\overline{\text{REFRESH}}$ is deasserted. RFCYC is ALE connected to the Am29C668 MCI input. Since RFCYC is normally High, the Am29C668 is normally in the read/write mode.

The system reset, RESET , is tied directly to the MC_0 input of the Am29C668. When RESET is asserted, the memory may be corrupted due to a violation of the DRAM timing requirements. It is assumed, for this application, that RESET indicates that the system is initializing or that an unrecoverable failure has occurred; therefore, the memory contents may be unreliable. If the application requires that memory be retained during system resets, a new signal MC_0 must be generated by the control logic. This signal goes active when RESET is asserted and the state machine is in the IDLE, PC1, PC2A or PC2B state. MC_0 must remain active until the state machine reaches the page-mode PM state. When RESET goes active, RFRQ is asserted and RFCYC is deasserted. The memory control logic asserts RASI and holds it asserted for four MCLOCK cycles. The falling edge of RASI, while MC_0 and MC_1 are High, resets the Am29C668 to the default configuration; all internal refresh counters are initialized to zero. The memory control logic continues to generate reset cycles until RESET is deasserted. The reset cycles also generate the "wake up" cycles to the DRAMs.

State PAL

The State PAL arbitrates between memory accesses and refresh cycles, generates the control signals to the Am29C668 and contains the state machine that controls the memory accesses. The following inputs are used:

MCLOCK	Inverted 40 MHz system clock
S_0, S_1	Status, from 80C286
M_IOL	Memory High, I/O Low, from 80C286
A[23]	Address bit 23, from 80C286
RESET	System Reset signal
CH	Cache Hit, from Am29C668
RFRQ	Refresh request signal, from the Interface PAL
RFCYC	Refresh Cycle, from the Interface PAL

The following outputs are generated:

$\text{STATE}[3:0]$	State variables for the state machine
RASI	Row Address Strobe Input, input to the Am29C668

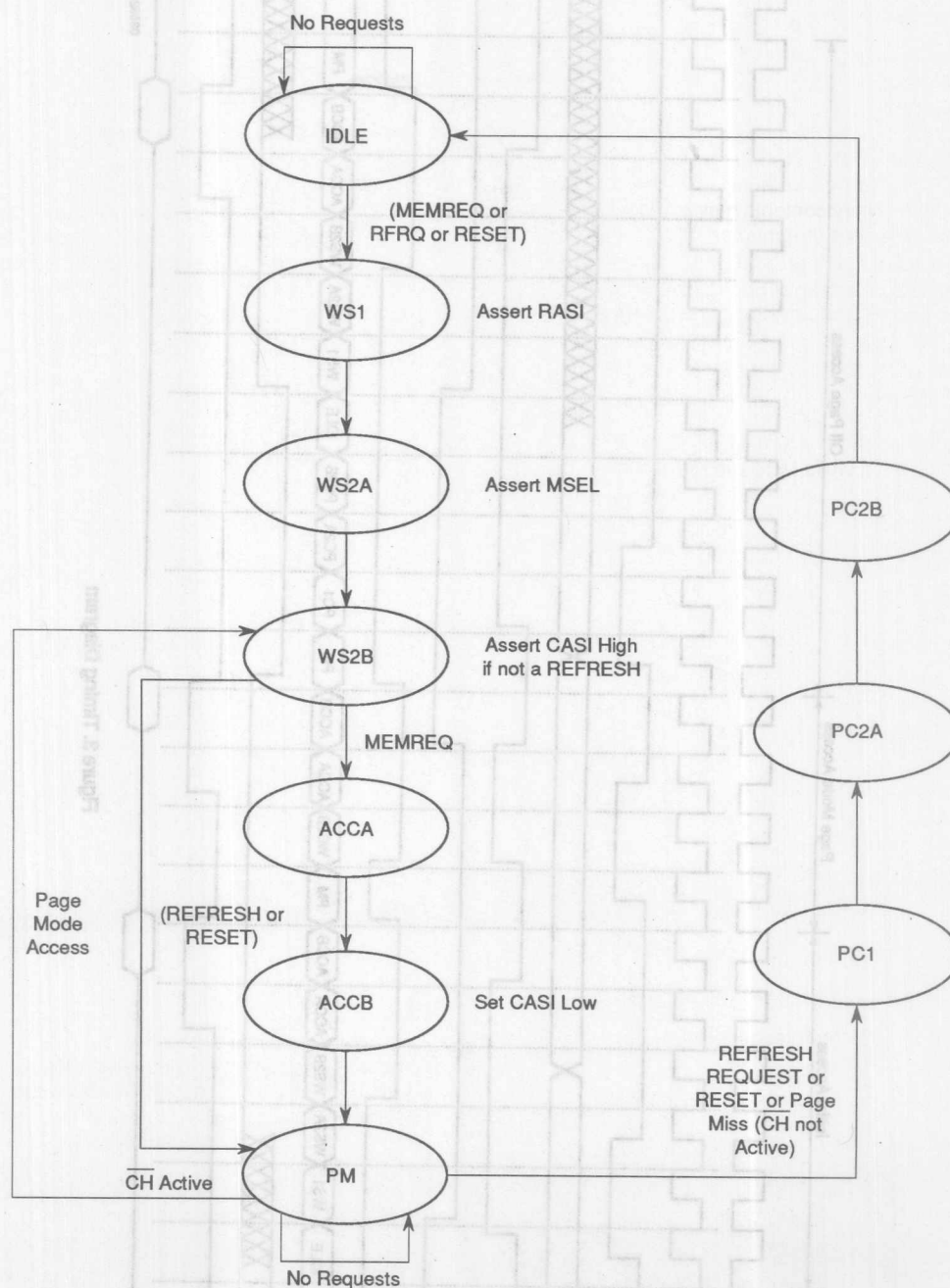
CASI	Column Address Strobe Input, input to the Am29C668
ALE	Address Latch Enable, input to the Am29C668
DELAY	Delay Cycle, used to synchronize memory accesses

The state variables $\text{STATE}[3:0]$ are used to keep track of the current status of the memory accesses. There are 16 states used, including six undefined states, as shown in the state diagram (Figure 2). If the state machine powers up in one of these six undefined states, the state machine goes to the IDLE state on the next clock. Figure 3 shows the timing of an initial access followed by a fast-page mode access and an off-page access.

When a valid memory request occurs, the state machine begins a long initial access. RASI is asserted and the state machine goes to the WS1 state. From the WS1 state, the controller unconditionally goes to WS2A, WS2B, ACCA and then ACCB. In the WS2A state, the MSEL input of the Am29C668 is asserted so that the column address can propagate to the DRAMs. This input is tied to the state machine output $\text{STATE}[1]$. In the WS2B state, CASI is asserted. These timings insure that the DRAM address setup and hold times are met relative to the falling edges of $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$.

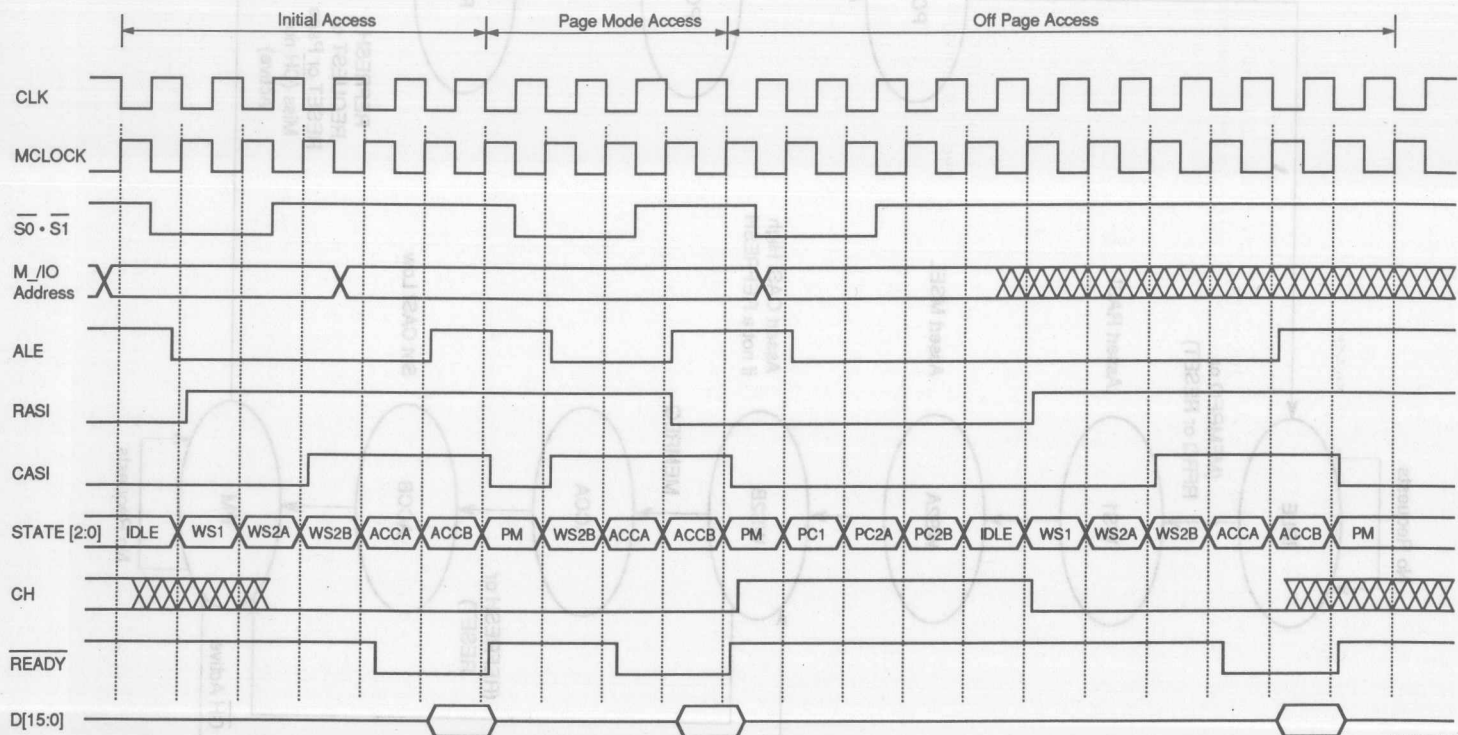
From the ACCB state, the memory controller goes to the PM state. The controller waits here until there is a refresh request or a valid memory request. If there is a memory request, CH is used to determine if the access is to the currently active page. If the access is to the same page, then the memory controller goes to the WS2B, ACCA and ACCB states, completing the access in just two cycles. If the access is outside the current page, a $\overline{\text{RAS}}$ precharge cycle must be performed followed by a long access. The $\overline{\text{RAS}}$ precharge cycle is performed by deasserting RASI and going from the PM state to PC1, PC2A, PC2B and then to the IDLE state. From the IDLE state, the memory controller performs a long initial access.

Refresh cycles perform the same accesses as memory accesses with a few exceptions. If a refresh request occurs while the memory controller is in the IDLE state, the refresh access waits one cycle to allow the row-refresh address adequate setup time relative to $\overline{\text{RAS}}$. The refresh access goes from the IDLE state to WS1, WS2A, WS2B states, then to the PM state. The memory controller must then perform a $\overline{\text{RAS}}$ precharge in the same manner it is performed during page-mode misses.



03552-002A

Figure 2. State Diagram



03552-003A

Figure 3. Timing Diagram

Sometimes, a memory access is requested during a refresh cycle. Since the refresh interval is controlled by the 555 timer, the refresh is asynchronous to the 80C286 memory requests. Under certain conditions, the refresh may start at the beginning of a bus cycle; all memory requests occur at the end of the bus cycle. If the 80C286 requests a memory access during this off-cycle refresh, the memory and the processor will be out of synchronization. This causes the memory controller to assert READY and data one cycle early and to remove them one cycle early. This situation is avoided by using the DELAY output to re-synchronize the memory and processor. If the memory request occurs during the wrong state, DELAY is asserted and the state machine delays one cycle. This re-synchronizes the state machine and the processor. The timing for re-synchronization is shown in Figure 4.

The ALE output from the State PAL is connected to the ALE input of the Am29C668; this signal controls the address latch of the CDMC. When ALE is High, the address latch is transparent and the address is latched on the falling edge of ALE. ALE goes Low when there is a valid memory request, and remains Low until the memory state machine is in the ACCA state, signaling that the memory access is about to complete. Asserting ALE High one cycle before the access completes provides enough time for the next address to be compared with the current address. This insures that CH is valid when it is evaluated by the state machine.

Refresh Cycles

To retain data, dynamic memories must be refreshed periodically to restore the charge on their storage capacitors. For 1-Mbit DRAMs, all 512 rows of memory must be refreshed every 8 ms. There are different methods to perform the refresh cycles: burst, forced and hidden; each has its advantages and disadvantages. The best method is determined by the instruction mix, system hardware and performance requirements.

The first method is burst-refresh cycles that refresh all 512 rows in one sequence. It works well in systems where there are long idle times between memory accesses. The main disadvantage of this system is an access to memory may be delayed for long periods during the refresh cycles, greatly impacting system throughput. This would definitely not be an acceptable method for real-time systems.

Another method is to periodically insert refresh cycles; this is called forced or distributed refresh. If refreshes are interspersed between memory accesses, the memory throughput and access time is not greatly impacted because there is a lower probability of refresh request and memory request contention. One refresh request is generated every $15.6 \mu\text{s} = 8 \text{ ms}/512 \text{ rows}$. This method is preferable to burst refresh in most systems.

It would be better if the refresh cycles occurred when the processor was accessing other memory or I/O. This type of refreshing is called hidden refresh since all or most of the refresh cycle is overlapped with another access. There are times, however, when the system continually accesses the same memory page and prevents the performance of hidden refreshes. If this happens, a forced refresh cycle must be performed. Hidden refreshing has the lowest system impact since all or most of the refresh cycle is overlapped with an access to another memory or I/O device. There are conceivable situations where hidden refresh would not perform as well as forced refresh. However, for most general applications, hidden refreshing offers the best performance.

This design utilizes forced refreshes instead of hidden refreshes. There are several reasons this method was selected. Additional logic is needed to keep track of when hidden refresh cycles are performed. This logic must suppress the forced refresh request after a hidden refresh cycle is performed and must force a refresh when no hidden refresh is performed. As a result, extra devices must be added that consume more board space, money and power.

A refresh cycle is identical to a normal access, except that the CAS outputs to the DRAMs are suppressed. The Am29C668 suppresses the CAS outputs in the refresh mode. REFRESH is asserted by the 555 timer every 10 μs to insure that the DRAMs maximum RAS active time is not violated. If a memory access is in process, the access is completed before the refresh cycle begins. If both a memory access and memory request occur during the IDLE state, the refresh request is given priority to insure that the refresh requirements of the DRAM are met.

555 Timer

A 555 timer is used to generate the refresh requests. The refresh period T is determined by the values of R1, R2 and C.

$$T = 0.693 [R1 + (2 \times R2)]C,$$

When R1 = 15 k Ω , R2 = 4.7 k Ω , and C = 470 pF,
T = 7.9 μs .

This meets the required refresh interval if 5% tolerance resistors and a 20% tolerance capacitor are used. Therefore, the maximum refresh overhead is 3.2%.

Normally the refresh interval for forced refresh is 15.8 μs . In this application, the refresh timer performs two functions: it generates the refresh request to maintain the data in the DRAMs and it times the RAS Low signal to guarantee that the maximum RAS Low time is not exceeded. The DRAMs have two specified maximum RAS Low times: t_{RAS} (10 μs) the maximum RAS Low

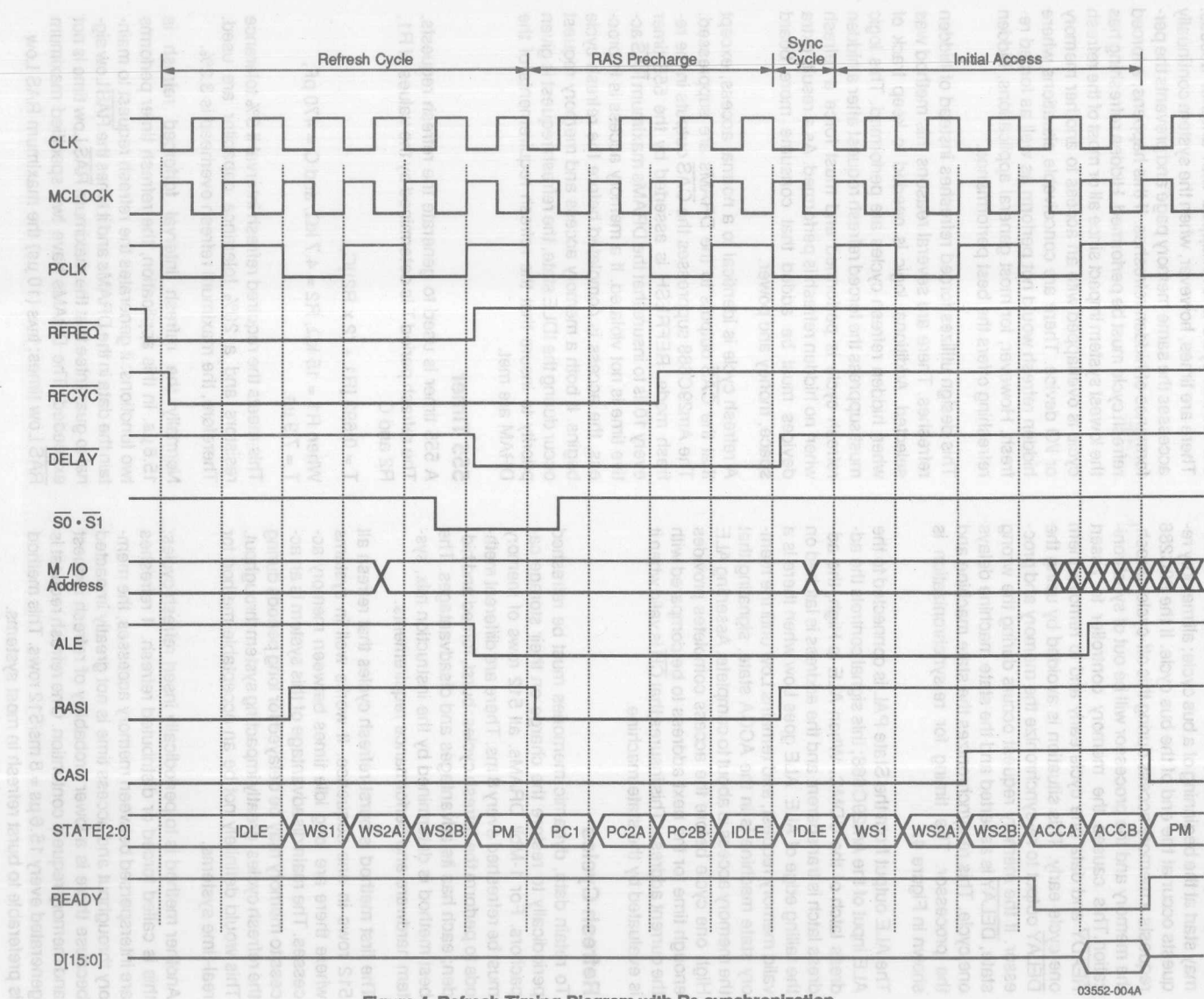


Figure 4. Refresh Timing Diagram with Re-synchronization

time during a normal access and t_{RASP} (100 μ s) the maximum \overline{RAS} Low time during fast-page-mode accesses. If the system can guarantee that every initial access is always followed by a fast-page-mode access, the refresh-timer interval can be 15.8 μ s. In most systems, however, this cannot be guaranteed. Therefore, the refresh timer must interrupt the memory every 10 μ s to guarantee that the DRAM t_{RAS} is not violated.

The 555 timer could be replaced with an external counter and logic to generate the refresh request. The counter could be synchronized to the processor so that all refresh requests terminate in the correct relationship to the processor clock. Another advantage of the counter implementation is that the exact refresh interval can be more precisely controlled, reducing the refresh overhead to a maximum of 2.5%. The major drawback is the increased cost and board space of the counter and logic over that of the 555 timer.

Data Buffers

Simple data buffers are used to minimize the propagation delay. Since the DRAMs have separate inputs and outputs, two 74F244s are used on the data input lines and two on the data output lines. The 74F244s on the data inputs are permanently enabled by tying their \overline{OE} inputs Low. The \overline{OE} inputs to the data output buffers are generated by the State PAL output \overline{OEB} .

Timing Analysis

There are two different DRAM parameters that must be examined to determine the access time. The first is \overline{RAS} -to- \overline{CAS} delay t_{RCD} . If t_{RCD} is greater than the specified $t_{RCD}(\max)$, the access is controlled by \overline{CAS} access time t_{CAC} . The second parameter is \overline{RAS} -to-column-address delay t_{RAD} . If t_{RAD} is greater than $t_{RAD}(\max)$, the access is controlled by the access time from column address t_{AA} . If both t_{RCD} and t_{RAD} are less than their specified maximums, the access time is determined by \overline{RAS} , t_{RAC} . If both t_{RCD} and t_{RAD} are greater than their maximums, the one yielding the slowest access time determines the memory access time.

In this design, MSEL is asserted one memory-clock cycle after RASI, and CASI is asserted two memory clock cycles after RASI (see Figure 5). The capacitive load on \overline{RAS}_n is approximately

$$7 \text{ pF/DRAM} \times 16 \text{ DRAMS} = 112 \text{ pF.}$$

For margin, assume 120 pF. The delay from RASI to \overline{RAS}_n is therefore,

$$26 \text{ ns} - (350 - 120) \text{ pF} \times 2.5 \text{ ns/50 pF} = 14.5 \text{ ns or approximately } 15 \text{ ns.}$$

The loading and delay for \overline{CAS}_n are the same; therefore, t_{RCD} is two memory-clock cycles or 50 ns. The load on the Q_n outputs is

$$5 \text{ pF/DRAM} \times 16 \text{ DRAMS/bank} \times 4 \text{ banks} = 320 \text{ pF.}$$

Assuming 350 pF for margin, the delay from MSEL to Q is 26 ns. Therefore,

$$t_{RAD} = 1 \text{ MCLOCK cycle} + \text{MSEL-to-}Q_n \text{ delay RASI-to-} \overline{RAS}_n \text{ delay} = 25 + 26 - 15 = 36 \text{ ns.}$$

Since $t_{RAD}(\max) = 40 \text{ ns}$ and $t_{RCD}(\max) = 60 \text{ ns}$, the access timing is determined by t_{RAC} . The access requires:

1 MCLOCK Cycle	25 ns
74ALS04 Inverter Delay	5 ns
MCLOCK to RASI	8 ns
RASI to \overline{RAS}_n	15 ns
DRAM Access	85 ns
Buffer Delay	7 ns
Data Setup	3 ns
Total Access Time	148 ns

With a total access time of 148 ns, the access is completed in three processor cycles (150 ns). Therefore, the State PAL must be a PAL1 6R8-10. MSEL is asserted on the cycle after RASI; this insures that the row-address hold time is met. CASI asserted one cycle after MSEL guarantees that the column-address set up time is met.

Consecutive accesses within the same page are limited by the \overline{CAS} access time:

1 MCLOCK Cycle	25 ns
74ALS04 Inverter	5 ns
MCLOCK to CASI	8 ns
CASI to \overline{CAS}_n	15 ns
DRAM Page Mode Access	30 ns
Buffer Delay	7 ns
Data Setup	3 ns
Total Access Time	93 ns

With a total access time of 93 ns, the access is completed DRAM and processor speeds. Table 2 gives the number in two cycles with a margin of $(2 \times 50) - 93 = 7$ ns of processor cycles required for fast-page-mode minimum. Table 1 summarizes the number of processor accesses for different DRAM and processor speeds. cycles required for the initial memory access for different DRAM and processor speeds. Table 2 gives the number of processor cycles required for fast-page-mode accesses for different DRAM and processor speeds.

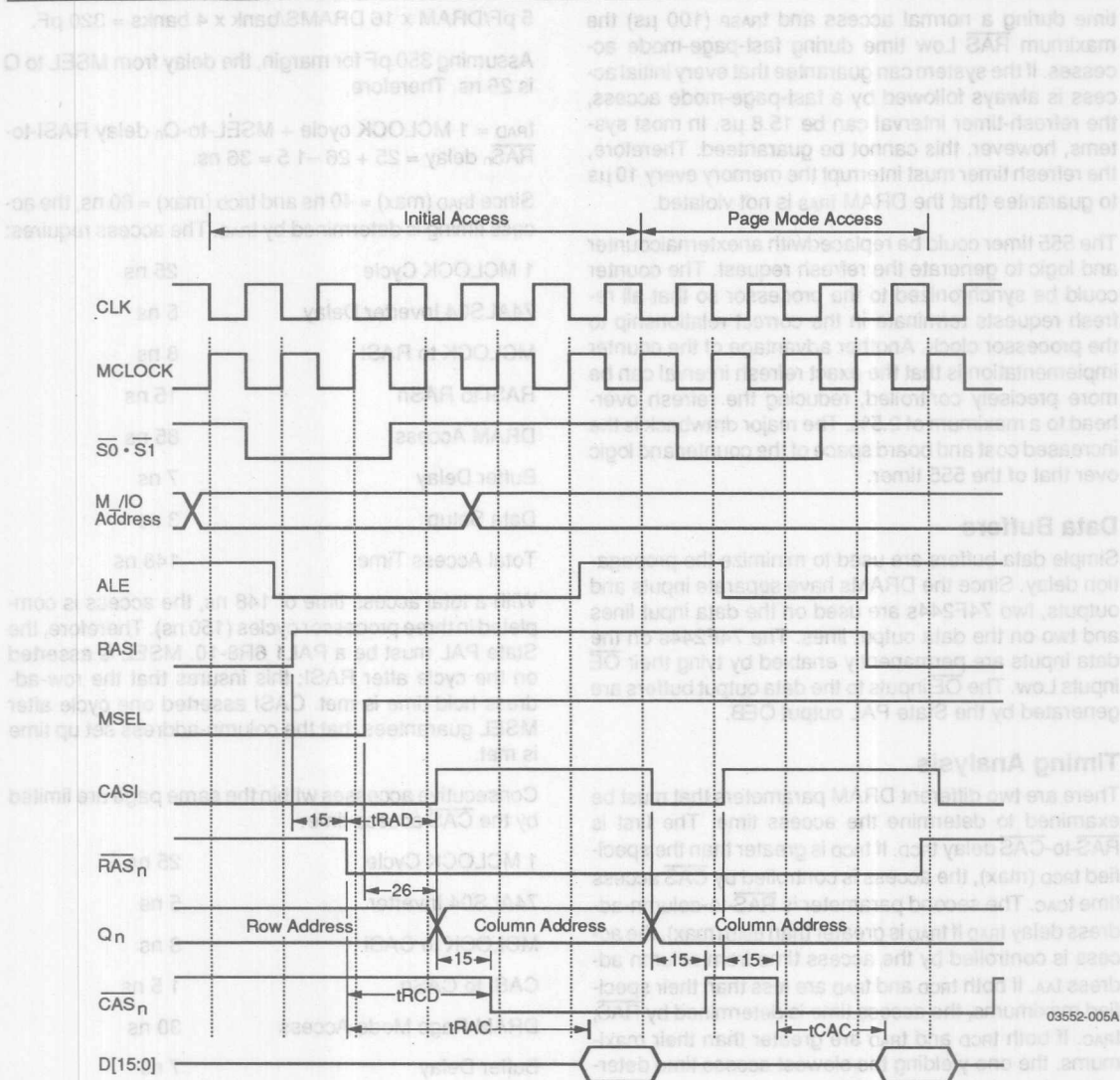


Figure 5. Timing Analysis

Table 1. Processor Cycles for Initial Access for Different Processor and Memory Speeds

Processor Speed (MHz)	Memory Speed (ns)			
	60	85	100	120
16	3	3	3	3
20	3	3	4	4
25*	3	4	4	5

* Assumes State PAL is PAL16R8-7.

Table 2. Processor Cycles for Page-Mode Access for Different Processor and Memory Speeds

Processor Speed (MHz)	Memory Speed (ns)*			
	60	85	100	120
16	2	2	2	2
20	2	2	2	2
25**	2	2	2	2

* 2-Cycle access requires zero wait state.

** Assumes State PAL is PAL16R8-7.

PARTS LIST

Part	Count
PALCE16V8	1
PALCE20V8	1
Am29C668	1
555 timer	1
74F244	4
Memories	64
20 MHz OSC	1
Total	73

PAL Equations

The following are the logic equations for the PALs. They are written in PALASM.

The following application notes are guides to interfacing the Am29C668 with popular microprocessors. They are paper designs only and have not been built and tested.

;PALASM Design Description

Declaration Segment

TITLE INTERFACE

PATTERN A

REVISION 1.0

COMPANY AMD INC.

DATE 07/12/91

CHIP _INT286 PALCE20V8

PIN Declarations

PIN 1	MCLOCK	COMBINATORIAL ; INPUT
PIN 2	/MSTATE[3]	COMBINATORIAL ; INPUT
PIN 3	/MSTATE[2]	COMBINATORIAL ; INPUT
PIN 4	/MSTATE[1]	COMBINATORIAL ; INPUT
PIN 5	/MSTATE[0]	COMBINATORIAL ; INPUT
PIN 6	RESET	COMBINATORIAL ; INPUT
PIN 7	/REFRESH	COMBINATORIAL ; INPUT
PIN 8	ALE	COMBINATORIAL ; INPUT
PIN 9	A[0]	COMBINATORIAL ; INPUT
PIN 10	/BHE	COMBINATORIAL ; INPUT
PIN 11	/S0	COMBINATORIAL ; INPUT
PIN 13	/OE	COMBINATORIAL ; INPUT
PIN 15	/WEH	COMBINATORIAL ; OUTPUT
PIN 16	/WEL	COMBINATORIAL ; OUTPUT
PIN 21	/RFDONE	REGISTERED ; OUTPUT
PIN 22	/READY	COMBINATORIAL ; OUTPUT
PIN 17	/OEB	REGISTERED ; OUTPUT
PIN 18	/RFCYC	REGISTERED ; OUTPUT
PIN 19	/RFRQ	REGISTERED ; OUTPUT
PIN 20	/RFINT	REGISTERED ; OUTPUT
PIN 12	GND	; INPUT
PIN 24	VCC	; INPUT

;SPECIAL DEFINITIONS

```

STRING IDLE_ST 'MSTATE[3] * MSTATE[2] * MSTATE[1] * /MSTATE[0]'
STRING WS1_ST 'MSTATE[3] * MSTATE[2] * MSTATE[1] * MSTATE[0]'
STRING WS2A_ST 'MSTATE[3] * MSTATE[2] * /MSTATE[1] * MSTATE[0]'
STRING WS2B_ST 'MSTATE[3] * /MSTATE[2] * /MSTATE[1] * MSTATE[0]'
STRING ACCA_ST 'MSTATE[3] * /MSTATE[2] * /MSTATE[1] * /MSTATE[0]'
STRING PM_ST ' /MSTATE[3] * /MSTATE[2] * /MSTATE[1] * MSTATE[0]'
STRING PC2B_ST 'MSTATE[3] * /MSTATE[2] * MSTATE[1] * /MSTATE[0]'

```



Boolean Equation Segment

EQUATIONS

READY = 1

RFINT := REFRESH * /RFDONE + RESET

RFRQ := RFINT + RFRQ * /(PM_ST * RFCYC)

RFCYC := IDLE_ST * RFRQ * /RFDONE * /RESET + PC2B_ST * RFRQ * /RFDONE * /RESET + RFCYC *
(WS1_ST + WS2A_ST + WS2B_ST + PM_ST)

RFDONE := RFCYC * WS1_ST + RFDONE * REFRESH * /RESET

WEL = /A[0] * ALE * S0 + WEL * /ALE * /RESET

WEH = BHE * ALE * S0 + WEH * /ALE * /RESET

OEB := /WEH * /WEL * /RFCYC * /RESET * (WS2A_ST + WS2B_ST + ACCA_ST)

Simulation Segment

SIMULATION

PALASM Design Description

Declaration Segment

TITLE STATE

PATTERN A

REVISION 1.0

COMPANY AMD INC.

DATE 07/12/91

CHIP _STATE286 PALCE16V8

PIN Declarations

```

PIN 1    MCLOCK    COMBINATORIAL ; INPUT
PIN 2    /S0       COMBINATORIAL ; INPUT
PIN 3    /S1       COMBINATORIAL ; INPUT
PIN 4    M_IOL     COMBINATORIAL ; INPUT
PIN 5    /CH       COMBINATORIAL ; INPUT
PIN 6    A[23]     COMBINATORIAL ; INPUT
PIN 7    /RFRQ     COMBINATORIAL ; INPUT
PIN 8    /RFCYC    COMBINATORIAL ; INPUT
PIN 9    RESET     COMBINATORIAL ; INPUT
PIN 11   /OE       COMBINATORIAL ; INPUT
PIN 12   /DELAY    REGISTERED ; OUTPUT
PIN 13   /ALE      REGISTERED ; OUTPUT
PIN 14   CASI      REGISTERED ; OUTPUT
PIN 15   /RASI     REGISTERED ; OUTPUT
PIN 16   /MSTATE[0] REGISTERED ; OUTPUT
PIN 17   /MSTATE[1] REGISTERED ; OUTPUT
PIN 18   /MSTATE[2] REGISTERED ; OUTPUT
PIN 19   /MSTATE[3] REGISTERED ; OUTPUT
PIN 10   GND       ; INPUT
PIN 20   VCC       ; INPUT

```

;SPECIAL DEFINITIONS

```

STRING IDLE_ST 'MSTATE[3] * MSTATE[2] * MSTATE[1] * MSTATE[0]'
STRING WS2A_ST 'MSTATE[3] * MSTATE[2] * MSTATE[1] * MSTATE[0]'
STRING ACCA_ST 'MSTATE[3] * MSTATE[2] * MSTATE[1] * MSTATE[0]'
STRING ACCB_ST 'MSTATE[3] * MSTATE[2] * MSTATE[1] * MSTATE[0]'
STRING PM_ST 'MSTATE[3] * MSTATE[2] * MSTATE[1] * MSTATE[0]'
STRING IDLE '#B1110'
STRING WS1 '#B1111'
STRING WS2A '#B1101'
STRING WS2B '#B1001'
STRING ACCA '#B1000'
STRING ACCB '#B0000'
STRING PM '#B0001'
STRING PC1 '#B0011'
STRING PC2A '#B0010'

```



```
STRING PC2B '#B1010'
```

```
STRING MEMREQ 'READ + WRITE'
```

```
STRING READ '/A[23] * M_IOL * S1'
```

```
STRING WRITE '/A[23] * M_IOL * S0'
```

```
;----- Boolean Equation Segment -----
```

```
EQUATIONS
```

```
ALE := MEMREQ + ALE * /(ACCA_ST * /RFCYC)
```

```
DELAY := MEMREQ * /ALE * (IDLE_ST * RFRQ * /RFCYC + MSTATE[1] * MSTATE[0] + MSTATE[3] *  
/MSTATE[2]) + DELAY * /(MSTATE[2] * /RFCYC)
```

```
CASE (MSTATE[3..0])
```

```
BEGIN
```

```
  IDLE:
```

```
    BEGIN
```

```
      IF (ALE * /DELAY + MEMREQ * /RFRQ + /RFCYC + RFRQ * RESET) THEN
```

```
        BEGIN
```

```
          MSTATE[3..0] := WS1
```

```
        END
```

```
      ELSE
```

```
        BEGIN
```

```
          MSTATE[3..0] := IDLE
```

```
        END
```

```
      END
```

```
    WS1:
```

```
      BEGIN
```

```
        MSTATE[3..0] := WS2A
```

```
      END
```

```
    WS2A:
```

```
      BEGIN
```

```
        MSTATE[3..0] := WS2B
```

```
      END
```

```
    WS2B:
```

```
      BEGIN
```

```
        IF (RFRQ) THEN
```

```
          BEGIN
```

```
            MSTATE[3..0] := PM
```

```
          END
```

```
        ELSE
```

```
          BEGIN
```

```
            MSTATE[3..0] := ACCA
```

```
          END
```

```
        END
```

```
    ACCA:
```

```
      BEGIN
```

```
        MSTATE[3..0] := ACCB
```

```

END
ACCB:
BEGIN
  MSTATE[3..0] := PM
END
PM:
BEGIN
  IF (RFCYC + RFRQ + /CH * MEMREQ) THEN
    BEGIN
      MSTATE[3..0] := PC1
    END
  ELSE
    BEGIN
      IF (CH * MEMREQ) THEN
        BEGIN
          MSTATE[3..0] := WS2B
        END
      ELSE
        BEGIN
          MSTATE[3..0] := PM
        END
      END
    END
  END
END
PC1:
BEGIN
  MSTATE[3..0] := PC2A
END
PC2A:
BEGIN
  MSTATE[3..0] := PC2B
END
PC2B:
BEGIN
  MSTATE[3..0] := IDLE
END
END ;"CASE"
CASI := WS2A_ST * /RFRQ + CASI * /ACCB_ST
RASI := IDLE_ST * /(ALE + MEMREQ * /RFRQ + RFCYC + RFRQ * RESET) + PM_ST * /(CH * MEMREQ + RFRQ)
+ /MSTATE[2] * MSTATE[1]
;----- Simulation Segment -----
SIMULATION
;

```

Am29C668 Configurable Dynamic Memory Controller to Am386™ Microprocessor Interface



INTRODUCTION

The interface between the Am29C668 4-Mbit Configurable Dynamic Memory Controller (CDMC) and the Am386 microprocessor was designed for maximum performance; therefore, lowering the total device count was a secondary concern. It is possible to interface the Am29C668 to the Am386 microprocessor with only one PAL®, but this would make many assumptions about the system implementation which are not generally applicable. This design is as general as possible so that the user may tailor his implementation to a specific memory system. Possible changes to the design are discussed with associated system requirements and implications. A block diagram, timing analyses and logic equations necessary to implement the design are included. This design requires a minimum number of external devices to perform the interface and glue functions: three PAL devices (one 20L8, one 22V10 and one 20X10A), one 74LS240 (six inverters) and four 74F245 bidirectional transceivers.

Distinctive Characteristics

- Am29C668 4-Mbit Configurable Dynamic Memory Controller/Driver with Auto Timing
- 16-MHz Am386 Microprocessor
- 120-ns Fast Page Mode 1 Mbit x 1 DRAM
- Two Wait-State Initial Accesses With Zero Wait-State Subsequent Page-Mode Accesses
- 4-Mbyte Dynamic Memory Expandable Up To 16 Mbyte per Am29C668
- Supports Up To 4 Gbytes of Physical Memory
- Supports Address Pipelining
- Performs Hidden Refresh Cycles to Maximize Memory Throughput

MEMORY ARCHITECTURE OVERVIEW

To obtain the maximum memory throughput but still maintain a reasonable cost, 120-ns fast page-mode DRAMs are used. The Am386 microprocessor requires a minimum of two processor cycles per access. If additional cycles are needed, the memory controller holds $\overline{\text{RDY}}$ inactive. The processor inserts wait states until $\overline{\text{RDY}}$ is asserted. The 16-MHz Am386 microprocessor completes the initial access to memory in two wait states (four cycles total). The subsequent accesses within the

page are performed with no wait states (two cycles total).

Page-mode DRAMs appear to the processor as if they are fast cache memories during page accesses. The page size for a 1-Mbit DRAM is 1024 bits or 1 Kbits. The memory is 32 bits wide, therefore the page size is 4 Kbytes. The Am29C668 detects accesses within the same page via on-chip cache-mode operation. When a new address is latched, it is compared with the previous address; if the row and bank addresses are the same (the upper 12 bits) then the Cache Hit signal $\overline{\text{CH}}$ is asserted. The memory state machine immediately begins the next access without deasserting $\overline{\text{RAS}}$. An access outside the page results in a five-cycle, three-wait-state access, two cycles for the $\overline{\text{RAS}}$ precharge and three for the data access. The actual performance enhancement of the memory depends upon the instruction mix of the program executed.

The memory array consists of four banks; each bank contains 4 Mbytes or 1 Mword (32 bits) of memory. This gives a maximum size of 16 Mbytes or 4 Mwords of memory.

A 16-MHz system was selected since high performance may be achieved even with relatively slow DRAMs. As processor speeds increase toward 20 and 25 MHz, the number of wait states increases until this memory architecture becomes impractical. Faster microprocessors demand faster DRAMs, static RAM caches, bank interleaving, or a combination of three or more other exotic architectures, implemented at considerable cost. These topics are beyond the scope of this application note.

FUNCTIONAL DESCRIPTION

The primary data paths and functional elements are shown in Figure 1. The following discussion describes each subsection of the block diagram, including the control logic, buffers and memory array.

Am29C668 CDMC

The Am29C668 generates the $\overline{\text{RAS}}$, $\overline{\text{CAS}}$ and address signals to the DRAM array; no external drivers are needed. Additionally, the Am29C668 generates the row addresses during refreshes from its internal refresh row-address counter. The Am29C668, operating in the auto-timing mode, generates the $\overline{\text{RAS}}$ -to- $\overline{\text{CAS}}$ and the $\overline{\text{RAS}}$ -to-address timing internally.



The Am29C668 must be programmed before any memory accesses can occur. This is accomplished through a dummy I/O access to the I/O address E190H. The control logic does not distinguish between I/O Reads and Writes to the CDMC configuration register. The input AC[10] must be tied Low to insure proper configuration. The address bits A[11:2] are written into the Am29C668 configuration register. The options selected are: a 4-bank RAS and CAS configuration, CAS byte decoding, RAS-only refresh, 1-Mbyte memory size, cache mode and auto timing.

The Am29C668 supports byte decoding through the CAsEN[3:0] inputs. Byte-enable outputs BE[3:0] from the Am386 are connected to these inputs and only the selected bytes are accessed. The unselected bytes perform a RAS-only refresh on the current row.

The auto-timing mode generates the RAS, CAS and multiplexed address inputs to the DRAMs. This timing is optimized for 100-ns DRAMs. The CASn outputs are further qualified by the CASIEN input (auto timing with external override). With RASi active, CASIEN is pulsed; this accesses the DRAM in fast-page mode. During these fast-page-mode accesses, the DRAMs are accessed with no wait states inserted. The CH signal from the Am29C668 is used to determine if the current address in the input latch has the same row as the previous address. If the fast-page-mode accesses are to the same bank and row, CH is asserted and a page mode access is initiated. If CH is deasserted, the RAS must be precharged and a normal access occurs.

Distinctive Characteristics

- Am29C668 4-Mbit Configurable Dynamic Memory Controller/Driver with Auto Timing
- 18-MHz Am386 Microprocessor
- 120-ns Fast Page Mode 1 Mbit x 1 DRAM
- Two Wait-State Initial Accesses With Zero Wait-State Subsequent Page-Mode Accesses
- 4-Mbyte Dynamic Memory Expandable Up To 16 Mbytes per Am29C668
- Supports Up To 4 Gbytes of Physical Memory
- Supports Address Pipelining
- Performs Hidden Refresh Cycles to Maximize Memory Throughput

MEMORY ARCHITECTURE OVERVIEW

To obtain the maximum memory throughput but still maintain a reasonable cost, 120-ns fast page-mode DRAMs are used. The Am386 microprocessor requires a minimum of two processor cycles per access. If additional cycles are needed, the memory controller holds RDY inactive. The processor inserts wait states until RDY is asserted. The 18-MHz Am386 microprocessor completes the initial access to memory in two wait states (four cycles total). The subsequent accesses within the

FUNCTIONAL DESCRIPTION

The primary data paths and functional elements are shown in Figure 1. The following discussion describes each subsection of the block diagram, including the control logic, buffers and memory array.

Am29C668 CDMC

The Am29C668 generates the RAS, CAS and address signals to the DRAM array; no external drivers are needed. Additionally, the Am29C668 generates the row addresses during refresh from its internal refresh-row address counter. The Am29C668, operating in the auto-timing mode, generates the RAS-to-CAS and the RAS-to-address timing internally.

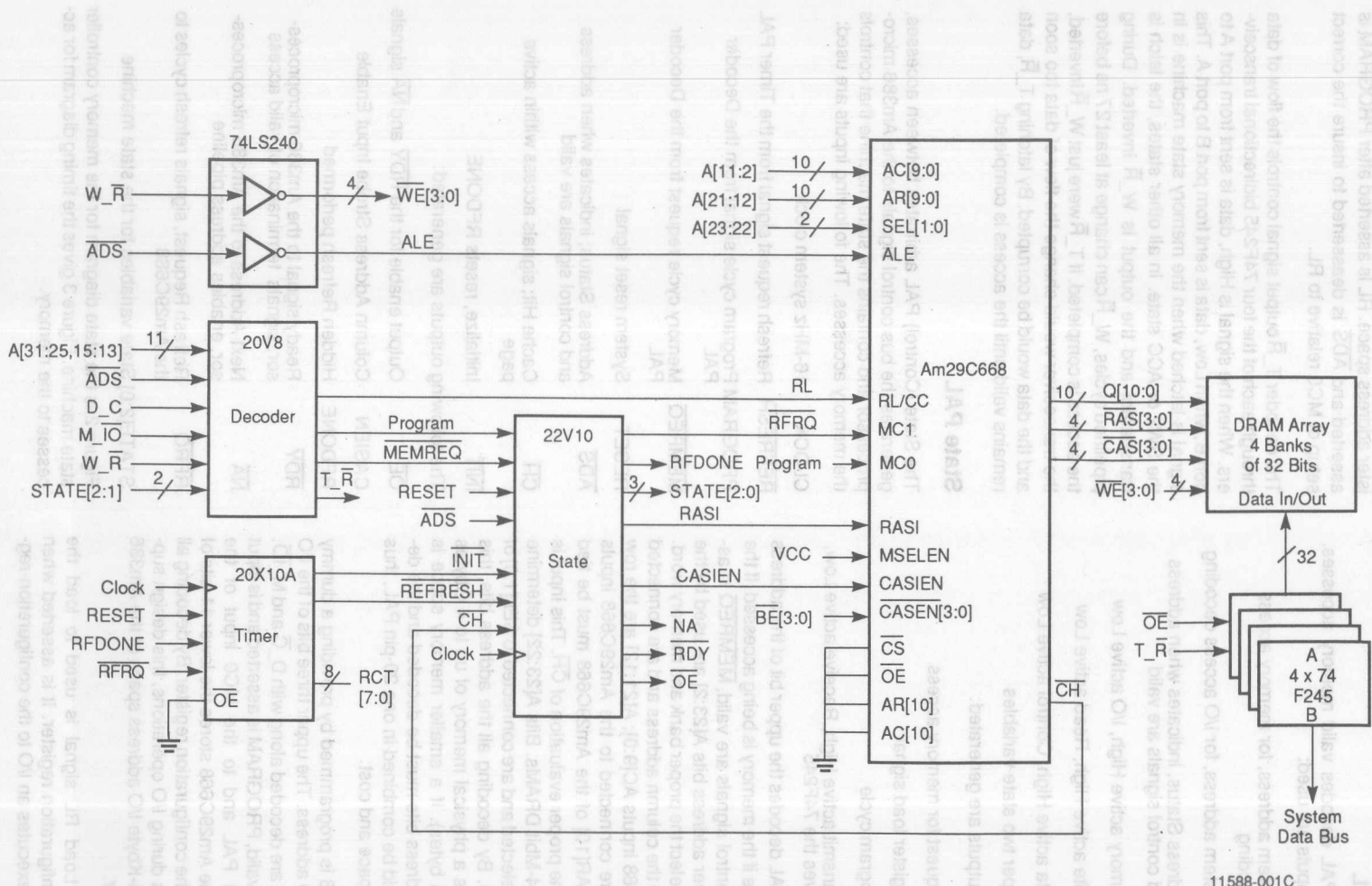


Figure 1. Interface Block Diagram

Decoder PAL

The Decoder PAL decodes valid memory accesses. The following inputs are used:

A[31:24]	System address, for memory access decoding
A[15:13]	System address, for I/O access decoding
\overline{ADS}	Address Status, indicates when address and control signals are valid
$M_{\overline{IO}}$	Memory active High, I/O active Low
$W_{\overline{R}}$	Write active High, Read active Low
$D_{\overline{C}}$	Data active High, Control active Low

STATE[2:1] Upper two state variables

The following outputs are generated:

\overline{MEMREQ}	Request for memory access
RL	Register load signal
PROGRAM	Program cycle
$T_{\overline{R}}$	Transmit active High, Receive active Low, drives the 74F245

The Decoder PAL decodes the upper bit of the address and $M_{\overline{IO}}$ to see if the memory is being accessed. If the address and control signals are valid, \overline{MEMREQ} is asserted. The lower address bits A[23:2] are used by the Am29C668 to select the proper bank and memory word. Bits A[11:2] are the column address and are connected to the Am29C668 inputs AC[9:0]; A[21:12] are the row address and are connected to the Am29C668 inputs AR[9:0]. Input AR[10] of the Am29C668 must be tied Low to insure the proper evaluation of \overline{CH} . This input is used only with 4-Mbit DRAMs. Bits A[23:22] determine which bank is selected and are connected to SEL[1:0] of the Am29C668. By decoding all the address bits, this design supports a physical memory of up to 4 Gbytes (4,294,967,296 bytes). If a smaller memory space is used, fewer address bits must be decoded and the decoder logic could be combined in one 20-pin PAL, thus saving board space and cost.

The Am29C668 is programmed by providing a dummy output to an I/O address. The upper three bits of the I/O space A[15:13] are decoded along with $D_{\overline{C}}$ and $M_{\overline{IO}}$. If the access is valid, PROGRAM is asserted and is input to the Control PAL and to the MC0 input of the Am29C668. The Am29C668 stores the lower 11 bits of the address in the configuration register. By decoding all 16 address bits during I/O operations, this design supports the full 64-Kbyte I/O address space of the Am386 microprocessor.

The Register Load RL signal is used to load the Am29C668 configuration register. It is asserted when the processor executes an I/O to the configuration-reg-

ister address space. RL is asserted after PROGRAM is asserted and \overline{ADS} is deasserted to insure the correct setup of MC0 relative to RL.

The decoder $T_{\overline{R}}$ output signal controls the flow of data through each of the four 74F245 bidirectional transceivers. When the signal is High, data is sent from port A to port B; when Low, data is sent from port B to port A. This signal is latched when the memory state machine is in the SW2 or ACC state. In all other states, the latch is transparent and the output is $W_{\overline{R}}$ inverted. During pipelined cycles, $W_{\overline{R}}$ can change at least 27 ns before the access is completed. If $T_{\overline{R}}$ were just $W_{\overline{R}}$ inverted, the transceiver would change the flow of data too soon and the data would be corrupted. By latching $T_{\overline{R}}$, data remains valid until the access is completed.

State PAL

The State (Control) PAL arbitrates between accesses, generates the bus control signals to the Am386 microprocessor and contains the state machine that controls the memory accesses. The following inputs are used:

CLOCK	16-MHz system clock
$\overline{REFRESH}$	Refresh request signal from the Timer PAL
PROGRAM	Program cycle signal from the Decoder PAL
\overline{MEMREQ}	Memory cycle request from the Decoder PAL
RESET	System reset signal
\overline{ADS}	Address Status; indicates when address and control signals are valid
\overline{CH}	Cache Hit; signals access within active page
INIT	Initialize, resets RFDONE

The following outputs are generated:

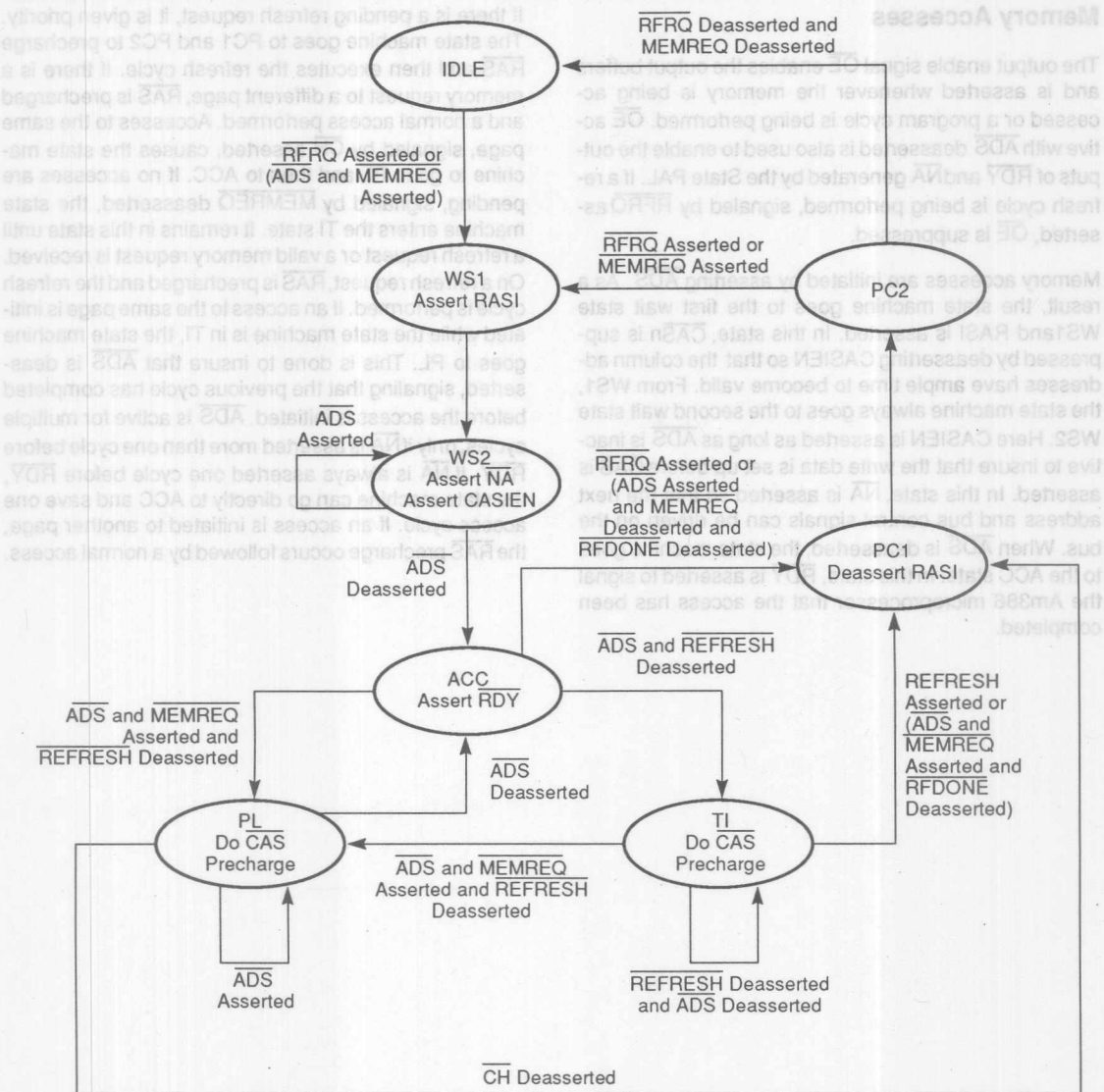
\overline{OE}	Output enable for the \overline{RDY} and \overline{NA} signals
CASIEN	Column Address Strobe Input Enable
RFDONE	Hidden Refresh performed
\overline{RDY}	Ready signal to the Am386 microprocessor, signals termination of valid access
\overline{NA}	Next Address to the Am386 microprocessor, enables address pipeline
\overline{RFRQ}	Refresh Request, signals refresh cycles to the Am29C668

STATE[2:0] State variables for the state machine

Figure 2 is the state diagram for the memory controller state machine. Figure 3 gives the timing diagram for accesses to the memory.

ated by asserting $\overline{\text{ADS}}$. As a goes to the first wait state. In this state, $\overline{\text{CAS}}_n$ is supplied so that the column address becomes valid. From WS1, goes to the second wait state asserted as long as $\overline{\text{ADS}}$ is inactive. Data is set up before $\overline{\text{CAS}}$ is asserted so that the next signals can be driven on the next state, the state machine goes to state, $\overline{\text{RDY}}$ is asserted to signal that the access has been

ADP
Dissolved



11588-002A

Figure 2. State Diagram

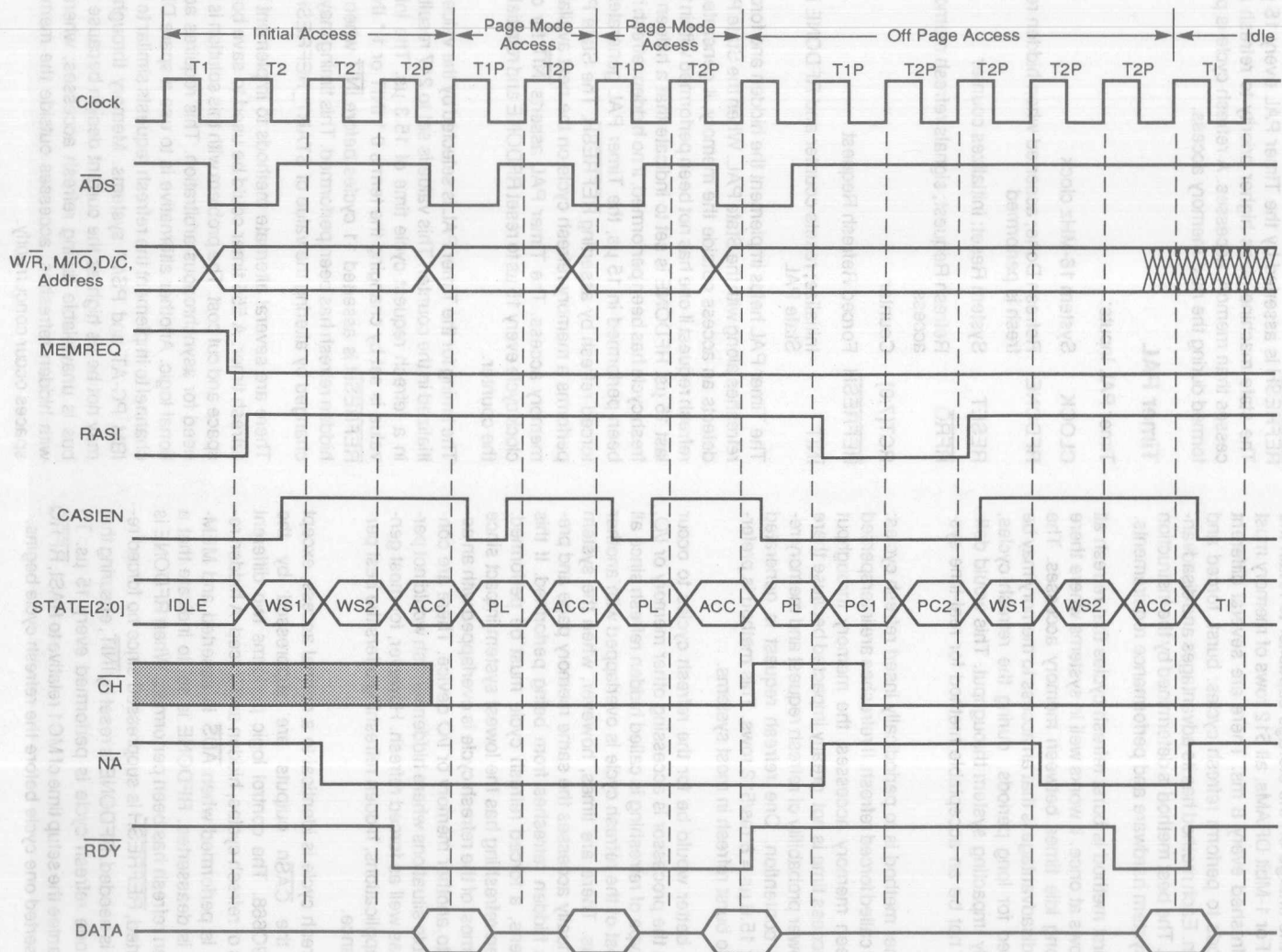


Figure 3. Timing Diagram

11588-003A

Refresh Cycles

Dynamic memories must be refreshed periodically to restore the charge on the storage capacitor to retain the data. For 1-Mbit DRAMs, all 512 rows of memory must be refreshed every 8 ms. There are several different methods to perform refresh cycles: burst, forced and hidden. Each method has its advantages and disadvantages. The best method is determined by the instruction mix, system hardware and performance requirements.

The first method is burst refresh cycles that refresh all 512 rows at once. It works well in systems where there are long idle times between memory accesses. The main disadvantage is that an access to memory may be delayed for long periods during the refresh cycles, greatly impacting system throughput. This would definitely not be an acceptable method for real-time systems.

Another method is to periodically insert refresh cycles; this is called forced refresh. If refreshes are interspersed between memory accesses, the memory throughput and access time is not greatly impacted because there is a lower probability of refresh request and memory request contention. One refresh request is generated every $15.6 \mu\text{s} = 8 \text{ ms}/512 \text{ rows}$. This method is preferable to burst refresh in most systems.

Even better would be for the refresh cycles to occur when the processor is accessing other memory or I/O. This type of refreshing is called hidden refresh since all or most of the refresh cycle is overlapped with another access. There are times, however, when the system continually accesses the same memory page and prevents hidden refreshes from being performed. If this happens, a forced refresh cycle must be performed. Hidden refreshing has the lowest system impact since all or most of the refresh cycle is overlapped with an access to another memory or I/O device. There are conceivable situations where hidden refresh would not perform as well as forced refresh. However, for most general applications, hidden refreshing offers the best performance.

A refresh cycle is identical to a normal access, except that the $\overline{\text{CAS}}_n$ outputs are suppressed by the Am29C668. The control logic performs two different types of refresh cycles, hidden and forced. A hidden refresh is performed when $\overline{\text{ADS}}$ is asserted and MEM-REQ is deasserted. RFDONE is set to indicate that a hidden refresh has been performed. When RFDONE is asserted, $\overline{\text{REFRESH}}$ is suppressed since no forced refresh is needed. RFDONE is reset by $\overline{\text{INIT}}$, ensuring that only one refresh cycle is performed every $15 \mu\text{s}$. To guarantee the setup time of MC1 relative to RASI , $\overline{\text{RFRQ}}$ is asserted one cycle before the refresh cycle begins.

There are times, however, when the system continually accesses the same memory and prevents hidden refreshes from being performed. If this happens, $\overline{\text{REFRESH}}$ is asserted by the Timer PAL every $15 \mu\text{s}$. The state machine gives higher priority to refresh accesses than memory accesses. A refresh cycle is performed during the next memory access.

Timer PAL

Timer PAL Inputs:

CLOCK	System 16-MHz clock
RFDONE	Refresh Done; asserted when hidden refresh is performed
RESET	System Reset; initializes counter
$\overline{\text{RFRQ}}$	Refresh Request; signals refresh memory access
$\text{RCT}[7:0]$	Counter
$\overline{\text{REFRESH}}$	Forced Refresh Request
$\overline{\text{INIT}}$	Initialize; resets counter and RFDONE in State PAL

The Timer PAL helps implement the hidden and forced refreshes along with the State PAL. When the State PAL detects an access outside the memory, it generates a refresh request if one has not been performed within the last $15 \mu\text{s}$. RFDONE is set to indicate that a hidden refresh cycle has been performed. If no hidden refresh has been performed in $15 \mu\text{s}$, the Timer PAL generates a forced refresh by asserting $\overline{\text{REFRESH}}$. The State PAL performs a memory refresh cycle on the next available memory access. The Timer PAL asserts $\overline{\text{INIT}}$ for one clock cycle every $15 \mu\text{s}$ to reset RFDONE and reinitialize the counter.

The timing for the Timer PAL is selected by the value initialized in the counter. This value is set to 247 resulting in a refresh request cycle time of $15.3 \mu\text{s}$. The initial value is set by changing the terms $0 * \overline{\text{INIT}}$ or $1 * \overline{\text{INIT}}$. $\overline{\text{REFRESH}}$ is asserted 11 cycles before $\overline{\text{INIT}}$ when no hidden refresh has been performed. This timing may be changed by altering the value of START_REFRESH .

There are several alternate methods to implement the refresh timer. A 555 timer could be used to save board space and cut cost. The problem with this solution is the need for asynchronous arbitration. This requires additional logic. Another alternative is to use a spare DMA channel to implement the refresh requests similar to the IBM PC-AT and PS/2* systems. Memory throughput may not be as high as the current design because the bus is unavailable during refresh accesses; whereas with hidden refreshes, accesses outside the memory spaces occur concurrently.

Data Buffers

Simple data buffers are used to minimize the propagation delay. The \overline{OE} is generated by the State PAL and $T_{\overline{R}}$ is generated by the Decoder PAL. Four 74F245s are used in this design since 32 bits are used.

Timing Analysis

The initial access to memory requires four processor cycles to complete. Each processor cycle is a minimum of 62 ns at 16 MHz. The hold time of the Am386 microprocessor does not affect timing shown here because it is hidden by the turn-off time of the data driver. The initial access requires:

T1 Cycle	62 ns
Clock to RASI	12 ns
RASI to \overline{RASn}	26 ns
DRAM Access	120 ns
Buffer Delay	7 ns
Data Setup	10 ns
Total Access Time	237 ns

With a total access time of 237 ns, the access is completed in four processor cycles with a $(4 \times 62) - 237 = 11$ ns margin. The second consecutive access requires:

$\frac{1}{2}$ T1 Cycle Precharge	31 ns
Clock Low to CASIEN	15 ns
CASIEN to \overline{CASn}	26 ns
DRAM Page Mode Access	30 ns
Buffer Delay	7 ns
Data Setup	10 ns
Total Access Time	119 ns

With a total access time of 119 ns, the access is completed in two cycles with a margin of $(2 \times 62) - 119 = 5$ ns. Non-consecutive accesses to the same page (an idle bus state between memory accesses) results in an extra wait state inserted. This wait state is necessary to insure that \overline{ADS} is deasserted, signaling a valid memory access, before the access is completed. If \overline{NA} is always asserted one cycle before \overline{RDY} , the access may be completed without the wait state because \overline{ADS} will always be deasserted before the completion of the access.

The \overline{RAS} precharge time for 120-ns DRAMs is 90 ns, therefore two processor cycles ($62 \times 2 = 124$ ns) are sufficient.

ALE is always valid 13 ns before the active High clock pulse. For four banks of 32 bits with parity, $36 \times 4 = 144$ DRAMs, the capacitive load on the address outputs will be $720 \text{ pF} = 5 \text{ pF/DRAM} \times 144 \text{ DRAMs}$. The Am29C668 requires 45 ns to drive the address lines; therefore the addresses are valid 32 ns after the active High clock pulse. This is sufficient to meet the address setup time to \overline{RAS} . CASIEN is not asserted until wait state WS_1 , allowing more than enough time for the column address to become valid. On consecutive accesses to the same page, the address is active 32 ns after the rising edge of the clock and this is more than 32 ns before the fastest \overline{CASn} .

The signal \overline{MEMREQ} must meet the setup time for the State PAL, which is 13 ns for the 22V10. Since the address is not valid until 40 ns after the rising edge of the clock, only 9 ns remain for the address decoding. This means that the Decoder PAL must be a 16L8-7 and the State PAL is a 22V10-15 in order to satisfy the system timings.

Refresh cycles require six processor cycles: two cycles for precharge and four cycles for the memory refresh. Table 1 shows the number of cycles needed to access memory for different processor and memory speeds.

Table 1. Access Cycles for Different Processor and Memory Speeds

Memory Speed (ns)	Processor Speed-ns					
	16 MHz		20 MHz		25 MHz	
	Initial	Cache Mode	Initial	Cache Mode	Initial	Cache Mode
120	4	2	5	3	6	3
100	4	2	5	3	5	3
85	4	2	4	3	5	3

PARTS LIST

Part	Count
PALCE20V8	1
PALCE22V10	1
PAL20X10A	1
Am29C668	1
74F245	4
74LS240	1
Memories	64
Total	73

PAL Equations

The following application notes are guides to interfacing the Am29C668 with popular microprocessors. They are paper designs only, and have not been built and tested.

The following are the logic equations for the three PAL devices. They are written in PALASM.

;PALASM Design Description

; Declaration Segment

TITLE DECODER

PATTERN A

REVISION 1.0

COMPANY AMD INC.

DATE 07/12/91

CHIP _DEC386 PALCE20V8

; PIN Declarations

PIN 1	A[31]	COMBINATORIAL ; INPUT
PIN 2	A[30]	COMBINATORIAL ; INPUT
PIN 3	A[29]	COMBINATORIAL ; INPUT
PIN 4	A[28]	COMBINATORIAL ; INPUT
PIN 5	A[27]	COMBINATORIAL ; INPUT
PIN 6	A[26]	COMBINATORIAL ; INPUT
PIN 7	A[25]	COMBINATORIAL ; INPUT
PIN 8	A[15]	COMBINATORIAL ; INPUT
PIN 9	A[14]	COMBINATORIAL ; INPUT
PIN 10	A[13]	COMBINATORIAL ; INPUT
PIN 11	A[12]	COMBINATORIAL ; INPUT
PIN 14	ADS	COMBINATORIAL ; INPUT
PIN 19	M_IOL	COMBINATORIAL ; INPUT
PIN 16	W_RL	COMBINATORIAL ; INPUT
PIN 17	MSTATE[2]	COMBINATORIAL ; INPUT
PIN 18	MSTATE[1]	COMBINATORIAL ; INPUT
PIN 23	D_CL	COMBINATORIAL ; INPUT
PIN 22	/MEMREQ	COMBINATORIAL ; OUTPUT
PIN 21	PROGRAM	COMBINATORIAL ; OUTPUT
PIN 20	T_RL	COMBINATORIAL ; OUTPUT
PIN 15	RL	COMBINATORIAL ; OUTPUT
PIN 12	GND	; INPUT
PIN 24	VCC	; INPUT

;SPECIAL DEFINITIONS

STRING PROGADDR 'A[15] * A[14] * A[13] * A[12]'

STRING MEMADDR '/A[31] * /A[30] * /A[29] * /A[28] * /A[27] * /A[26] * /A[25]'

; Boolean Equation Segment

EQUATIONS

MEMREQ = M_IOL * MEMADDR

PROGRAM = /M_IOL * PROGADDR * D_CL

RL = ADS * PROGRAM

T_RL = /W_RL * /(MSTATE[2] * MSTATE[1]) + T_RL * /MSTATE[2] * MSTATE[1]

; Simulation Segment

SIMULATION

;PALASM Design Description

;

Declaration Segment

TITLE TIMER

PATTERN A

REVISION 1.0

COMPANY AMD INC.

DATE 07/12/91

CHIP _TIMER386 PAL20X10

PIN Declarations

```

PIN 1    CLOCK      COMBINATORIAL ; INPUT
PIN 2    RFDONE      COMBINATORIAL ; INPUT
PIN 3    RESET       COMBINATORIAL ; INPUT
PIN 4    /RFRQ       COMBINATORIAL ; INPUT
PIN 13   OE          COMBINATORIAL ; INPUT
PIN 14   /INIT       REGISTERED ; OUTPUT
PIN 15   /REFRESH    REGISTERED ; OUTPUT
PIN 16   RCT[0]       REGISTERED ; OUTPUT
PIN 17   RCT[1]       REGISTERED ; OUTPUT
PIN 18   RCT[2]       REGISTERED ; OUTPUT
PIN 19   RCT[3]       REGISTERED ; OUTPUT
PIN 20   RCT[4]       REGISTERED ; OUTPUT
PIN 21   RCT[5]       REGISTERED ; OUTPUT
PIN 22   RCT[6]       REGISTERED ; OUTPUT
PIN 23   RCT[7]       REGISTERED ; OUTPUT
PIN 12   GND         ; INPUT
PIN 24   VCC         ; INPUT

```

;SPECIAL DEFINITIONS

STRING START_REFRESH '/RCT[7] * /RCT[6] * /RCT[5] * /RCT[4] * RCT[3]

* /RCT[2] * RCT[1] * RCT[0]'

STRING ONE 'VCC'

STRING ZERO 'GND'

Boolean Equation Segment

EQUATIONS

```

INIT := /RCT[7] * /RCT[6] * /RCT[5] * /RCT[4] * /RCT[3] * /RCT[2] * /RCT[1] * /RCT[0] + RESET
REFRESH := /RFDONE * START_REFRESH + REFRESH * /RFRQ
RCT[0] := /RCT[0] * /INIT + INIT * ONE
RCT[1] := (/INIT * /RCT[0]) :+ (/INIT * RCT[1] + INIT * ONE)
RCT[2] := (/INIT * /RCT[0] * /RCT[1]) :+ (/INIT * RCT[2] + INIT * ZERO)
RCT[3] := (/INIT * /RCT[0] * /RCT[1] * /RCT[2]) :+ (/INIT * RCT[3] + INIT * ONE)
RCT[4] := (/INIT * /RCT[0] * /RCT[1] * /RCT[2] * /RCT[3]) :+ (/INIT * RCT[4] + INIT * ONE)
RCT[5] := (/INIT * /RCT[0] * /RCT[1] * /RCT[2] * /RCT[3] * /RCT[4]) :+ (/INIT * RCT[5] + INIT * ONE)
RCT[6] := (/INIT * /RCT[0] * /RCT[1] * /RCT[2] * /RCT[3] * /RCT[4] * /RCT[5]) :+ (/INIT * RCT[6] + INIT * ONE)
RCT[7] := (/INIT * /RCT[0] * /RCT[1] * /RCT[2] * /RCT[3] * /RCT[4] * /RCT[5] * /RCT[6]) :+ (/INIT * RCT[7] +
INIT * ONE)

```

Simulation Segment

SIMULATION

;

;PALASM Design Description

Declaration Segment

TITLE STATE

PATTERN A

REVISION 1.0

COMPANY AMD INC.

DATE 07/12/91

CHIP _STATE386 PAL22V10

PIN Declarations

```

PIN 1    CLOCK    COMBINATORIAL ; INPUT
PIN 2    ADS      COMBINATORIAL ; INPUT
PIN 3    /REFRESH COMBINATORIAL ; INPUT
PIN 4    PROGRAM  COMBINATORIAL ; INPUT
PIN 5    CH       COMBINATORIAL ; INPUT
PIN 6    /MEMREQ  COMBINATORIAL ; INPUT
PIN 7    RESET    COMBINATORIAL ; INPUT
PIN 8    INIT     COMBINATORIAL ; INPUT
PIN 14   CASIEN   COMBINATORIAL ; OUTPUT
PIN 15   /RDY     REGISTERED ; OUTPUT
PIN 16   /RFRQ    REGISTERED ; OUTPUT
PIN 17   MSTATE[2] REGISTERED ; OUTPUT
PIN 18   MSTATE[1] REGISTERED ; OUTPUT
PIN 19   MSTATE[0] REGISTERED ; OUTPUT
PIN 20   RASI     REGISTERED ; OUTPUT
PIN 21   /NA      REGISTERED ; OUTPUT
PIN 22   RFDONE   REGISTERED ; OUTPUT
PIN 23   OE       COMBINATORIAL ; OUTPUT
PIN 12   GND      ; INPUT
PIN 24   VCC      ; INPUT

```

;SPECIAL DEFINITIONS

STRING IDLE '/MSTATE[2] * /MSTATE[1] * /MSTATE[0]'

STRING WS1 '/MSTATE[2] * /MSTATE[1] * MSTATE[0]'

STRING WS2 '/MSTATE[2] * MSTATE[1] * /MSTATE[0]'

STRING ACC '/MSTATE[2] * MSTATE[1] * MSTATE[0]'

STRING PL 'MSTATE[2] * MSTATE[1] * MSTATE[0]'

STRING TI 'MSTATE[2] * MSTATE[1] * /MSTATE[0]'

STRING PC1 'MSTATE[2] * /MSTATE[1] * /MSTATE[0]'

STRING PC2 'MSTATE[2] * /MSTATE[1] * MSTATE[0]'

STRING IDLE_ST '#B000'

STRING WS1_ST '#B001'

STRING WS2_ST '#B010'

STRING ACC_ST '#B011'

STRING PL_ST '#B111'

STRING TI_ST '#B110'

STRING PC1_ST '#B100'

STRING PC2_ST '#B101'

Boolean Equation Segment

EQUATIONS

OE = (WS1 + WS2 + ACC + PL + PROGRAM) * /RFRQ

CASIEN = RASI * /RFRQ * (/ADS * WS1 + ADS + WS2 + ACC + PL * /CLOCK)

IF (RESET) THEN

BEGIN

MSTATE[2..0] := IDLE_ST

RFRQ := 0

RASI := 0

RFDONE := 0

END

CASE (MSTATE[2..0])

BEGIN

IDLE_ST:

BEGIN

```

IF (RFRQ * /PROGRAM) THEN
  BEGIN
    MSTATE[2.0] := WS1_ST
    RFRQ := 1
    RASI := 1
    RFDONE := RFDONE * /INIT
  END
ELSE
  BEGIN
    IF (ADS * MEMREQ * /REFRESH) THEN
      BEGIN
        MSTATE[2.0] := WS1_ST
        RFDONE := RFDONE * /INIT
        RASI := 1
      END
    ELSE
      BEGIN
        RASI := 0
        MSTATE[2.0] := WS2_ST
        IF (PROGRAM * ADS * /REFRESH) THEN
          BEGIN
            RDY := 1
            RFRQ := (ADS * /PROGRAM * /MEMREQ * /RFDONE) + REFRESH
            RFDONE := RFDONE * /INIT
          END
        END
      END
    END
  END
WS1_ST:
  BEGIN
    MSTATE[2.0] := WS2_ST
    RFRQ := RFRQ
    RFDONE := RFRQ * /REFRESH * /INIT
    RASI := 1
    IF (/RFRQ * ADS) THEN
      BEGIN
        NA := 1
      END
    END
  END
WS2_ST:
  BEGIN
    RASI := 1
    RFRQ := RFRQ
    RFDONE := RFDONE * /INIT
    IF (ADS) THEN
      BEGIN
        MSTATE[2.0] := ACC_ST
        IF (/RFRQ) THEN
          BEGIN
            NA := 1
            RDY := 1
          END
        END
      END
    END
  END
ACC_ST:
  BEGIN
    RFRQ := RFRQ
    RFDONE := RFDONE * /INIT
    IF (RFRQ+REFRESH+(ADS * CH * MEMREQ)+(ADS * /MEMREQ * /RFDONE)) THEN
      BEGIN
        MSTATE[2.0] := PC1_ST
      END
    END
  END

```



```

RASI := 0
END
ELSE
BEGIN
  IF (ADS * CH * MEMREQ) THEN
    BEGIN
      MSTATE[2..0] := PL_ST
      NA := 1
      RASI := 1
      RDY := 1
    END
  ELSE
    BEGIN
      IF (ADS + (ADS * /MEMREQ)) THEN
        BEGIN
          MSTATE[2..0] := TI_ST
          RASI := 1
        END
      END
    END
  END
END
PL_ST:
BEGIN
  RFRQ := RFRQ
  NA := 1
  RASI := 1
  RFDONE := RFDONE * /INIT
  IF (ADS) THEN
    BEGIN
      MSTATE[2..0] := ACC_ST
      RDY := 1
    END
  ELSE
    BEGIN
      MSTATE[2..0] := PL_ST
    END
  END
END
TI_ST:
BEGIN
  RFDONE := RFDONE * /INIT
  RFRQ := RFRQ
  IF (REFRESH + (ADS * MEMREQ * CH) + (ADS * /MEMREQ * /RFDONE)) THEN
    BEGIN
      MSTATE[2..0] := PC1_ST
      RASI := 0
    END
  ELSE
    BEGIN
      IF (ADS * MEMREQ * CH) THEN
        BEGIN
          MSTATE[2..0] := PL_ST
          NA := 1
          RASI := 1
        END
      ELSE
        BEGIN
          MSTATE[2..0] := TI_ST
          RASI := 1
        END
      END
    END
  END
END

```

```
PC1_ST:
  BEGIN
    MSTATE[2..0] := PC2_ST
    RASI := 0
    RFRQ := REFRESH + /MEMREQ * /RFDONE * /RFRQ
    RFDONE := RFDONE * /INIT
  END
PC2_ST:
  BEGIN
    RFRQ := RFRQ
    RFDONE := RFDONE * /INIT
    IF (/RFRQ * /MEMREQ) THEN
      BEGIN
        MSTATE[2..0] := IDLE_ST
        RASI := 0
      END
    ELSE
      BEGIN
        MSTATE[2..0] := WS1_ST
        RASI := 1
      END
    END
  END
END ;"CASE"
;----- Simulation Segment -----
SIMULATION
;
```

```

PC1_ST
BEGIN
MSTATE2.0] = PC2_ST
RASI := 0
RFRQ := REFRESH + MEMREQ * VPDONE * RFRQ
RPDONE := RPDONE * VINT
END
PC2_ST
BEGIN
RFRQ := RFRQ
RPDONE := RPDONE * VINT
IF (RFRQ * MEMREQ) THEN
BEGIN
MSTATE2.0] = IDLE_ST
RASI := 0
END
ELSE
BEGIN
MSTATE2.0] = W21_ST
RASI := 1
END
END
END "CASE"
SIMULATION
Simulation Segment

```

Am29C668 Configurable Dynamic Memory Controller to 68020 Microprocessor Interface



INTRODUCTION

The interface between the Am29C668 4 Mbit Configurable Dynamic Memory Controller (CDMC) and the Motorola 68020 microprocessor was designed for maximum performance; therefore, lowering the total device count was a secondary concern. It is possible to interface the Am29C668 to the 68020 with fewer PAL® devices, but this would make many assumptions about the system implementation which are not generally applicable. This design is as general as possible so that the user may tailor his implementation to a specific memory system. Possible changes to the design are discussed with associated system requirements and implications. A block diagram, timing analyses and logic equations necessary to implement the design are included. This design requires a minimum number of external devices to perform the interface and glue functions: three PAL devices (one 16L8, one 22V10 and one 20X10), a decoder, and four 74F245 bidirectional transceivers.

Distinctive Characteristics

- Am29C668 4-Mbit Configurable Dynamic Memory Controller/Driver with Auto Timing
- 20-MHz 68020 Microprocessor
- 120-ns Fast Page Mode 1 Mbit x 1 DRAMs
- Two Wait-State Initial Accesses With Zero Wait-State Subsequent Page-Mode Accesses
- 4-Mbyte Dynamic Memory Expandable Up To 8-Mbyte per Am29C668

MEMORY ARCHITECTURE OVERVIEW

To obtain the maximum memory throughput but still maintain a reasonable cost, 120-ns fast page-mode 1-Mbit DRAMs are used. The 68020 requires a minimum of three processor cycles per access. If additional cycles are needed, the memory controller holds $\overline{\text{DSACK}}[1:0]$ inactive. The processor inserts wait states until $\overline{\text{DSACK}}[1:0]$ are asserted. The 20 MHz 68020 completes the initial access to memory in two wait states (five cycles total). The subsequent accesses within the page are performed with no wait states (three cycles total).

Page-mode DRAMs appear to the processor as if they are fast cache memories during page accesses. The page size for a 1-Mbit DRAM is 1024 bits or 1 Kbits. The 120-ns page-mode DRAM is 32 bits wide, therefore the page size is 4 Kbytes. The Am29C668 detects accesses within the same page via on-chip cache-mode operation. When a new address is latched, it is compared with the previous address; if the row and bank addresses are the same, the Cache Hit signal $\overline{\text{CH}}$ is asserted. The memory state machine immediately begins the next access. An access outside the page, a page miss, causes the memory controller to perform the $\overline{\text{RAS}}$ precharge for the DRAMs. The total access time on a page miss requires seven cycles, one for decoding, two cycles for the $\overline{\text{RAS}}$ precharge and four for the data access. This method of accessing memory results in shorter access times than memories using normal DRAM accesses. For certain systems, this memory can result in near-zero wait-state accesses. Only static RAMs can guarantee zero-wait-state accesses. The actual performance of the memory depends upon the instruction mix of the programs executed.

The memory array consists of two banks, each containing 4 Mbytes or 1 Mword (32 bits) of memory. This gives a maximum size of 8 Mbytes or 2 Mwords of memory per Am29C668 controller.

A 20-MHz system was selected since high performance may be achieved even with 120-ns DRAMs. As processor speeds increase toward 25 and 30 MHz, the number of wait states increases until this memory architecture becomes impractical. Faster microprocessors demand faster DRAMs, static RAM caches, bank interleaving, specialty mode DRAMs, or more exotic architectures, implemented at considerable cost. These topics are beyond the scope of this application note.

FUNCTIONAL DESCRIPTION

The primary data paths and functional elements are shown in Figure 1. The following discussion describes each subsection of the block diagram, including the control logic, buffers and memory array.



Figure 1. Interface Block Diagram.

Am29C668 CDMC

The Am29C668 generates the $\overline{\text{RAS}}$, $\overline{\text{CAS}}$ and address signals to the DRAM array; no external drivers are needed. Additionally, the Am29C668 generates the row addresses during $\overline{\text{RAS}}$ -only refreshes from its internal refresh row-address counter. The Am29C668, operating in the auto-timing mode, generates the $\overline{\text{RAS}}$ -to- $\overline{\text{CAS}}$ and the $\overline{\text{RAS}}$ -to-address timing internally.

The Am29C668 must be programmed before any memory accesses can occur. This is accomplished through a dummy memory access because the 68020 memory maps all I/O devices. The Am29C668 occupies a 4-Kbyte or 1-Kword address space. The actual decoding is left up to the user since it is highly system dependent. The address bits A[11:2] contain the value to be loaded into the configuration register. For this design, the lower 12 address bits are 198H. The options selected are: two banks $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$ configuration, $\overline{\text{CAS}}$ byte decoding, $\overline{\text{RAS}}$ -only refresh, 1-Mbit DRAM size, cache mode and auto-timing. If the part is used in only one configuration, only a single byte of address space is needed. The input AC[10] must be tied Low to place the Am29C668 in normal-mode operation. AC[10] is only used with 4-Mbit DRAMs.

When the Am29C668 is configured for two banks, $\overline{\text{RAS}}[1:0]$ are connected to the $\overline{\text{RAS}}$ inputs of bank 0, and $\overline{\text{RAS}}[3:2]$ are connected to bank 1 to minimize the capacitive load each output must drive. $\overline{\text{CAS}}[3]$ controls byte 3 of both banks and similarly $\overline{\text{CAS}}[2]$ controls byte 2; $\overline{\text{CAS}}[1]$ controls byte 1 and $\overline{\text{CAS}}[0]$ controls byte 0. This provides for byte accesses to memory.

The Am29C668 supports byte decoding through the $\overline{\text{CAsEN}}[3:0]$ inputs. The two least significant address bits A[1:0] and the Size bits $\text{SiZ}[1:0]$ from the 68020 are decoded to generate four byte-enable signals, $\overline{\text{BE}}[3:0]$. These byte-enable signals are connected to $\overline{\text{CAsEN}}[3:0]$ and only the selected bytes are accessed. The bytes that are not selected perform a $\overline{\text{RAS}}$ -only refresh on the current row address. (Note: $\overline{\text{BE}}[3]$ enables bits 31 to 24, $\overline{\text{BE}}[2]$ enables bits 23 to 16, $\overline{\text{BE}}[1]$ enables bits 15 to 8, $\overline{\text{BE}}[0]$ enables bits 7 to 0.)

The auto-timing mode generates the $\overline{\text{RAS}}$, $\overline{\text{CAS}}$ and multiplexed address inputs to the DRAMs. This timing is optimized for 100-ns DRAMs. The $\overline{\text{CH}}$ signal from the

Am29C668 is used to determine if the current address in the input latch has the same row and bank addresses as the previous access. If the access is to the active row, $\overline{\text{CH}}$ is asserted and a page-mode access is initiated. If $\overline{\text{CH}}$ is deasserted, $\overline{\text{RAS}}$ must be precharged and a normal access occurs.

The Address Strobe signal $\overline{\text{AS}}$ from the 68020 is input directly to the Address Latch Enable ALE of the Am29C668. When $\overline{\text{AS}}$ is deasserted, the address latch of the Am29C668 is transparent. When $\overline{\text{AS}}$ is asserted, the address is latched.

Byte Decoder PAL

The Byte Decoder PAL generates the byte-enable, write-enable, register load and output-enable signals. The following inputs are used:

A[1:0]	System address bit 1 and 0, for byte decoding.
SiZ[1:0]	Size from 68020, indicates the number of bytes remaining to be transferred.
$\overline{\text{AS}}$	Address Strobe, signals valid address and control signals.
$\overline{\text{DS}}$	Data Strobe, indicates valid data on the bus during write cycles, or signals memory to drive data on the bus during read cycles.
R_W	Read active High, Write active Low, from 68020.
PROGRAM	Program cycle, from decoder.
$\overline{\text{MEMREQ}}$	Memory Access Request, from decoder.

The following outputs are generated:

$\overline{\text{WE}}[1:0]$	Write Enable, enables writing to DRAMs.
$\overline{\text{BE}}[3:0]$	Byte Enable, selects active bytes during access.
$\overline{\text{DSACKEN}}$	Data and Size Acknowledge Enable, enables $\overline{\text{DSACK}}$ and bus drivers during valid memory accesses and programming cycles.
RL	Register load signal, input to Am29C668.

Table 1. Byte Enable Decoding for the 68020

A1	A0	SIZ1	SIZ0	BE3	BE2	BE1	BE0	Access Type
0	0	0	1	1	1	1	0	Byte
0	1	0	1	1	1	0	1	
1	0	0	1	1	0	1	1	
1	1	0	1	1	1	1	1	
0	0	1	0	1	1	0	0	Word
0	1	1	0	1	0	0	1	
1	0	1	0	0	0	1	1	
1	1	1	0	0	1	1	1	
0	0	1	1	1	0	0	0	3 Bytes
0	1	1	1	0	0	0	1	
1	0	1	1	0	0	1	1	
1	1	1	1	0	1	1	1	
0	0	0	0	0	0	0	0	Long Word
0	1	0	0	0	0	0	1	
1	0	0	0	0	0	1	1	
1	1	0	0	0	1	1	1	

Note:

0 is 0 V and 1 is 5 V.

The Write Enable $\overline{WE}[1:0]$ signals are input to the DRAM array to control read and write accesses. When active, a write access is performed and when inactive, a read access is performed. One signal is generated per bank to reduce the capacitive load on the output driver. All write cycles performed are "early" write cycles (\overline{WE} active before \overline{CAS}) that prevent the DRAM output driver from turning on. As a result, the Data In DIN and Data Out DOUT pins of the DRAMs can be tied together, reducing the number of routes on the PC board.

Byte Enable outputs $\overline{BE}[3:0]$ are used to select the bytes to be accessed and to control the assertion of the \overline{CAS}_n outputs. They are generated from the address bits $A[1:0]$ and the $SIZ[1:0]$ inputs from the 68020. Table 1 gives this decoding scheme.

The Register Load RL signal is used to load the Am29C668 configuration register. It is asserted when the processor performs an access to the configuration-register address space. RL is asserted after PROGRAM is asserted and \overline{AS} is asserted to ensure the correct setup of MC0.

State PAL

The State PAL arbitrates between memory accesses and refresh cycles, generates the bus-control signals to the 68020 and contains the state machine that controls the memory accesses. The following inputs are used:

CLOCK	20-MHz system clock.
REFRESH	Refresh request signal from the Timer PAL.
PROGRAM	Program cycle signal from the Decoder PAL.
MEMREQ	Memory cycle request from the Decoder PAL.
RESET	System reset signal.
\overline{AS}	Address Strobe, indicates when address signals are valid.
\overline{DS}	Data Strobe, indicates when data is valid.
CH	Cache Hit; signals access within active page.

The following outputs are generated:

RASI	Row Address Strobe Input, input to the Am29C668.
PGHIT	PGHIT, signals when an access to the same page is detected.
DSACKEN	DSACK Enable, enables \overline{DSACK} output.
$\overline{DSACK}[1:0]$	Data Size and Acknowledge signal to the 68020, signals termination of valid access and size of memory port.

RFRQ Refresh Request, signals refresh cycles to the Am29C668.

STATE[2:0] State variables for the state machine.

Data and Size Acknowledge Enable $\overline{DSACKEN}$ is output to the four 74F245s and to the state PAL. This signal enables the output drivers of the bus transceivers. $\overline{DSACKEN}$ also enables the output \overline{DSACK} . This signal is necessary since $\overline{DSACK}[1:0]$ are asserted during both program and memory accesses. During program cycles, the output buffers are enabled, but the data is not written to memory.

$\overline{DSACK}[1:0]$ are generated by the state PAL so that the memory can drive both inputs Low to signal the processor that it is a 32-bit device and that the memory access is complete. $\overline{DSACK}[1:0]$ are enabled by $\overline{DSACKEN}$ so that multiple drivers can be connected to these inputs. $\overline{DSACK}[1:0]$ must be pulled up through a resistor to maintain the proper logic levels when not being driven.

CASIEN is used as an external override in the auto-timing mode. This input is also used during fast-page mode accesses. CASIEN is strobed by \overline{DS} , causing the \overline{CAS}_n outputs to pulse, and thereby access the DRAMs. \overline{DS} is ANDed with \overline{MEMREQ} to generate CASIEN. \overline{DS} is used to guarantee that the data is valid before \overline{CAS}_n is asserted during Write cycles.

Figure 2 is the state diagram for the memory-controller state machine. Figure 3 gives the timing diagram for accesses to the memory. The following is a list of the state-machine states:

IDLE	Idle State, waiting for memory request or refresh request.
SW1	System Wait 1, memory access in progress.
SW2	System Wait 2, memory access in progress.
ACC	Access cycle, data valid during this cycle.
PM	Page Mode Wait, waiting for memory request or refresh request with RASI asserted.
PC1	Precharge cycle 1, \overline{RAS} Precharge cycle 1.
PC2	Precharge cycle 2, \overline{RAS} Precharge cycle 2.
PC12	Precharge cycle 1 or 2, \overline{RAS} Precharge cycle 1 or 2 depending upon the type of access.

Memory Accesses

Memory accesses are initiated by the 68020. The 68020 drives a valid address and a read/write signal onto the bus. Address Strobe \overline{AS} is then asserted by the 68020 to signal that a memory access has begun; it is valid a maximum of 25 ns after the clock goes Low. Therefore, there is no setup time for \overline{AS} relative to the clock. This means that RASI must be an asynchronous signal. RASI is not asserted until the clock is active to ensure that there is always one full cycle before the state machine begins. \overline{AS} is connected directly to the ALE input of the Am29C668.

Once RASI is asserted, the state machine proceeds through states SW1, SW2 and finally ACC where the data is valid. If a refresh access is pending, the state machine goes to state PC1 on the next cycle. The memory performs the RAS precharge cycles, PC1 and PC2, and then performs the refresh access. If no refresh request is pending, the memory goes to state PM.

In the PM state, RASI is still asserted, but the \overline{CAS}_n outputs are disabled because the \overline{CASIEN}_n inputs are deasserted. When \overline{AS} is asserted, the address of the current access is compared with the previous access. If the access is to the same row, the Am29C668 asserts \overline{CH} . The state machine goes to state ACC and the access is completed in three cycles, no wait states inserted. If the access is to another page, a page miss, the state machine goes to the PC12 state. The state machine then goes to the IDLE state to guarantee the \overline{RAS} precharge time; a normal access is then completed.

If $\overline{REFRESH}$ is asserted while the state machine is in the PM state, the state machine goes to PC12. From PC12, it goes to the PC2 state, thus completing the \overline{RAS} precharge. The state machine then goes directly to the SW1 state, completes a normal access and the \overline{RAS} precharge cycle. After a refresh cycle, the memory always goes to the IDLE state.

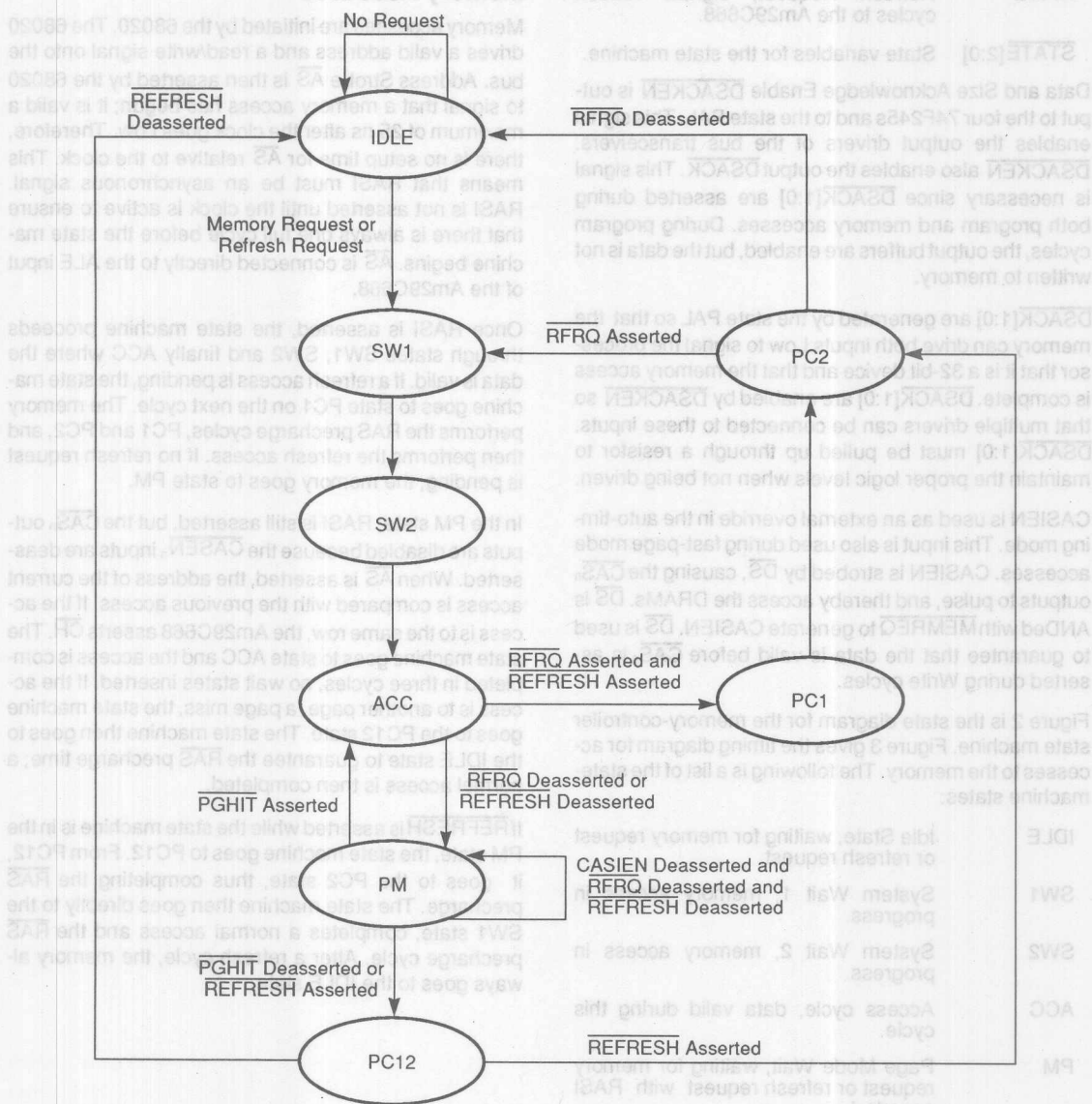
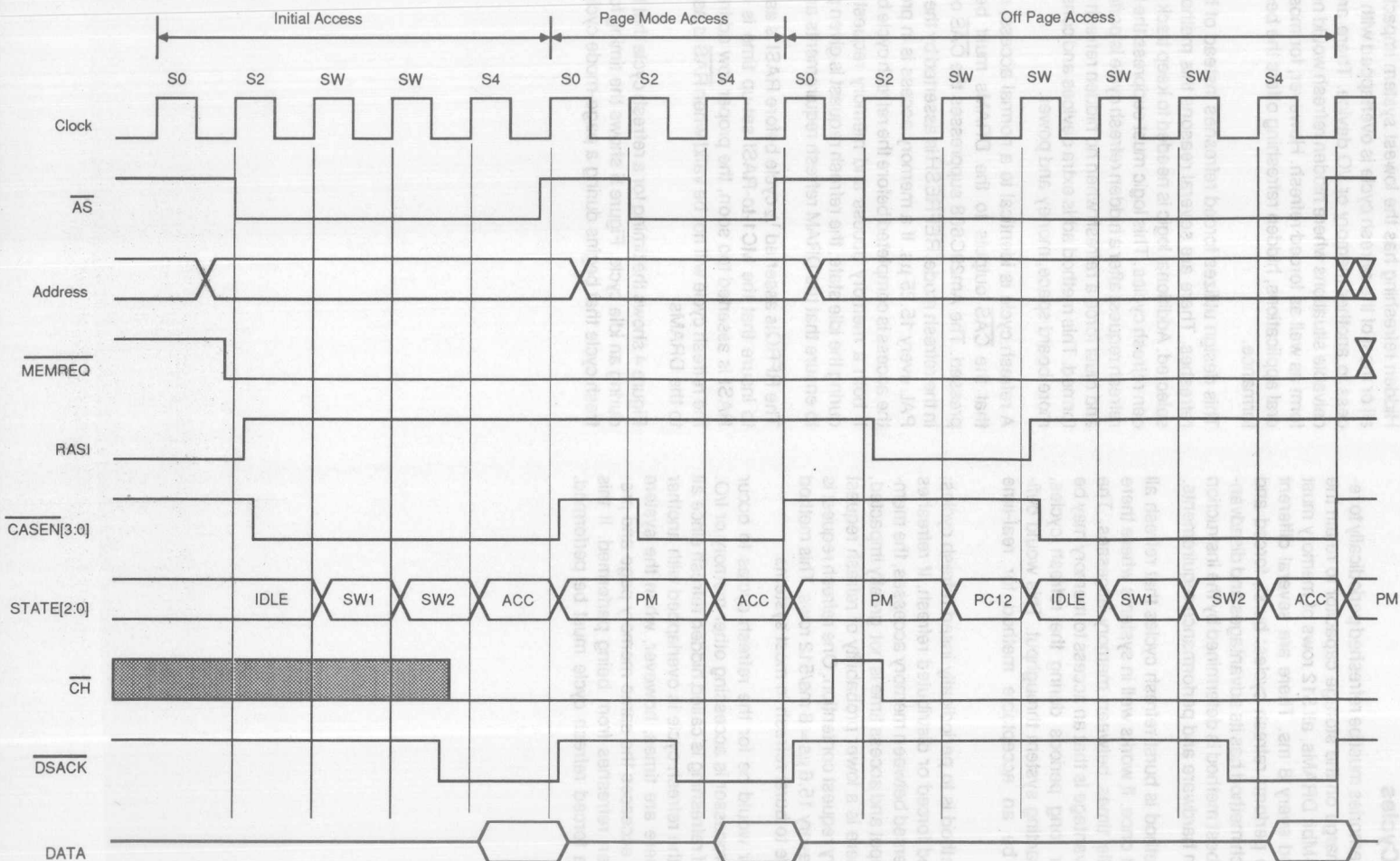


Figure 2. State Diagram

11865-002A



11588-003A

Figure 3. Timing Diagram

Refresh Cycles

Dynamic memories must be refreshed periodically to restore the charge on the storage capacitor to retain the data. For 1-Mbit DRAMs, all 512 rows of memory must be refreshed every 8 ms. There are several different methods to perform refresh cycles: burst, forced and hidden. Each method has its advantages and disadvantages. The best method is determined by the instruction mix, system hardware and performance requirements.

The first method is burst refresh cycles that refresh all 512 rows at once. It works well in systems where there are long idle times between memory accesses. The main disadvantage is that an access to memory may be delayed for long periods during the refresh cycles, greatly impacting system throughput. This would definitely not be an acceptable method for real-time systems.

Another method is to periodically insert refresh cycles; this is called forced or distributed refresh. If refreshes are interspersed between memory accesses, the memory throughput and access time is not greatly impacted, because there is a lower probability of refresh request and memory request contention. One refresh request is generated every $15.6 \mu\text{s} = 8 \text{ ms}/512 \text{ rows}$. This method is preferable to burst refresh in most systems.

Even better would be for the refresh cycles to occur when the processor is accessing other memory or I/O. This type of refreshing is called hidden refresh since all or most of the refresh cycle is overlapped with another access. There are times, however, when the system continually accesses the same memory page and prevents hidden refreshes from being performed. If this happens, a forced refresh cycle must be performed.

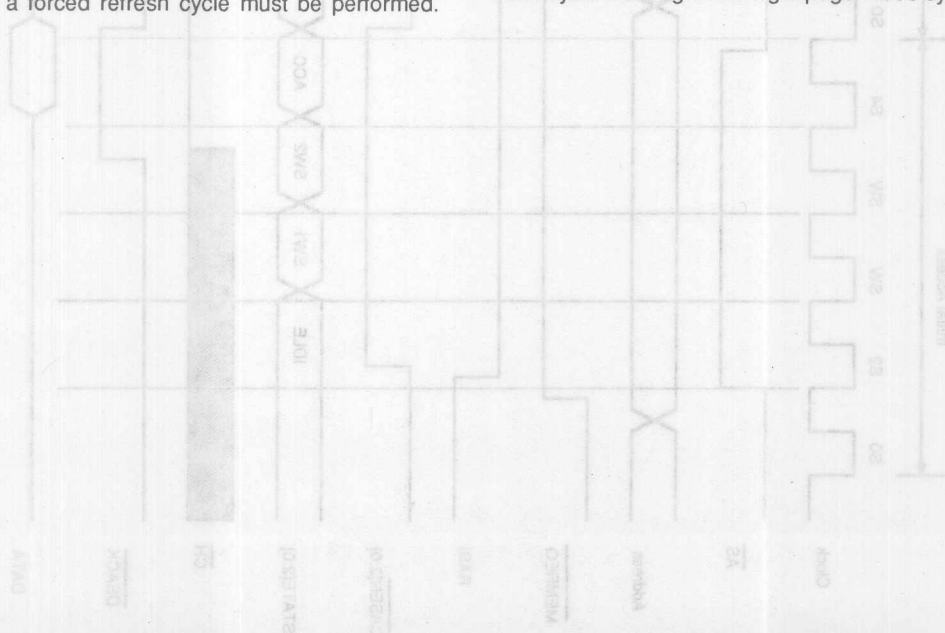
Hidden refreshing has the lowest system impact since all or most of the refresh cycle is overlapped with an access to another memory or I/O device. There are conceivable situations where hidden refresh would not perform as well as forced refresh. However, for most general applications, hidden refreshing offers the best performance.

This design utilizes forced refreshes instead of hidden refreshes. There are several reasons this method was selected. Additional logic is needed to keep track of hidden refresh cycles. This logic must suppress the forced refresh request after a hidden refresh cycle is performed and must force a refresh when no hidden refresh is performed. This method adds extra devices and consumes more board space, money and power.

A refresh cycle is identical to a normal access, except that the $\overline{\text{CAS}}$ outputs to the DRAMs must be suppressed. The Am29C668 suppresses the $\overline{\text{CAS}}$ outputs in the refresh mode. $\overline{\text{REFRESH}}$ is asserted by the Timer PAL every $15.25 \mu\text{s}$. If a memory access is in process, the access is completed before the refresh cycle begins. If both a memory access and memory request occur during the idle state, the refresh request is given priority to ensure that the DRAM refresh requirements are met.

The $\overline{\text{RFRQ}}$ is asserted $1/2$ cycle before $\overline{\text{RAS}}$ is asserted to insure that the MC1-to-RAS1 set-up time is met. If $\overline{\text{RAS}}$ is asserted too soon, the proper row address for the refresh cycle will not be valid when $\overline{\text{RAS}}$ is asserted to the DRAMs.

Figure 4 shows the timing for a refresh cycle that begins during an idle cycle. Figure 5 shows the timing for a refresh cycle that begins during a page-mode cycle.



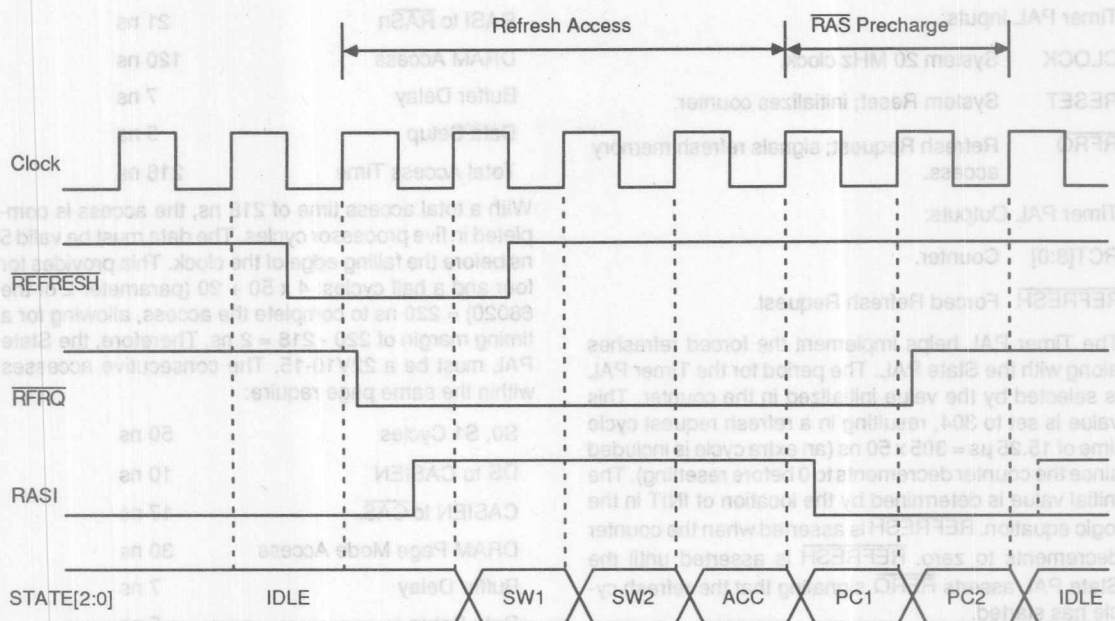


Figure 4. Refresh Timing from Initial Idle State

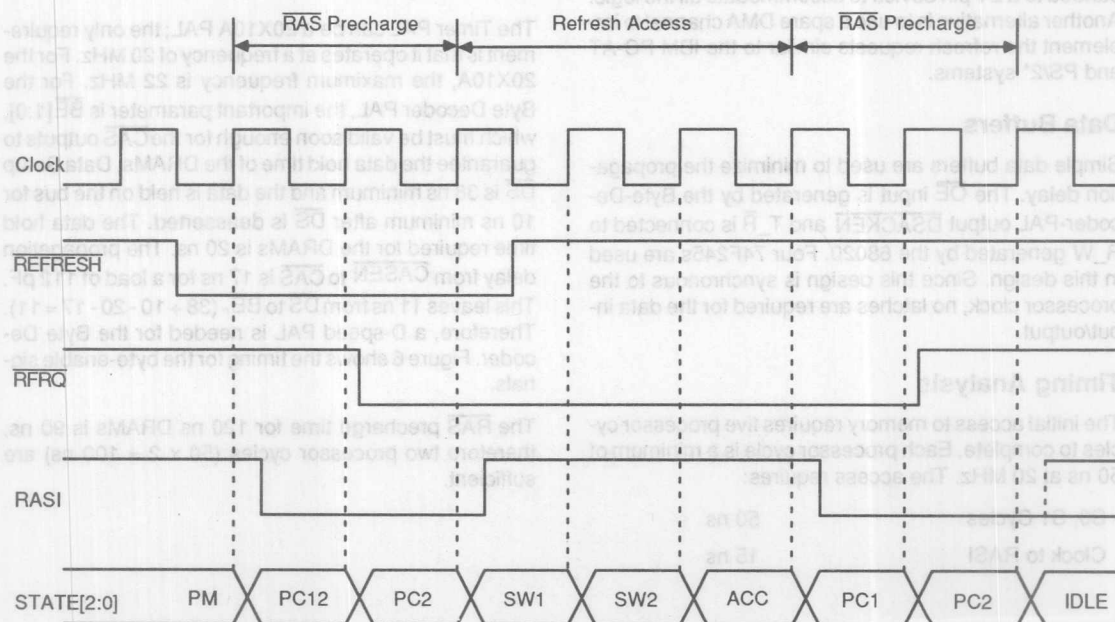


Figure 5. Refresh Timing from Initial Page Mode State

Timer PAL

Timer PAL Inputs:

CLOCK	System 20 MHz clock.	RAS _I to $\overline{\text{RAS}}_n$	21 ns
RESET	System Reset; initializes counter.	DRAM Access	120 ns
$\overline{\text{RFRQ}}$	Refresh Request; signals refresh memory access.	Buffer Delay	7 ns
		Data Setup	5 ns
		Total Access Time	218 ns

Timer PAL Outputs:

RCT[8:0]	Counter.
$\overline{\text{REFRESH}}$	Forced Refresh Request.

The Timer PAL helps implement the forced refreshes along with the State PAL. The period for the Timer PAL is selected by the value initialized in the counter. This value is set to 304, resulting in a refresh request cycle time of $15.25 \mu\text{s} = 305 \times 50 \text{ ns}$ (an extra cycle is included since the counter decrements to 0 before resetting). The initial value is determined by the location of INIT in the logic equation. $\overline{\text{REFRESH}}$ is asserted when the counter decrements to zero. $\overline{\text{REFRESH}}$ is asserted until the State PAL asserts $\overline{\text{RFRQ}}$ signaling that the refresh cycle has started.

There are several alternate methods to implement the refresh timer. A 555 timer could be used to save board space and cut cost. The problem with this solution is the need for asynchronous arbitration. The necessary logic could be included in the Byte Decoder PAL if it is expanded to a 24-pin device to accommodate all the logic. Another alternative is to use a spare DMA channel to implement the refresh requests similar to the IBM PC-AT and PS/2* systems.

Data Buffers

Simple data buffers are used to minimize the propagation delay. The $\overline{\text{OE}}$ input is generated by the Byte Decoder-PAL output $\overline{\text{DSACKEN}}$ and T_R is connected to R_W generated by the 68020. Four 74F245s are used in this design. Since this design is synchronous to the processor clock, no latches are required for the data input/output.

Timing Analysis

The initial access to memory requires five processor cycles to complete. Each processor cycle is a minimum of 50 ns at 20 MHz. The access requires:

S0, S1 Cycles	50 ns
Clock to RAS _I	15 ns

With a total access time of 218 ns, the access is completed in five processor cycles. The data must be valid 5 ns before the falling edge of the clock. This provides for four and a half cycles: $4 \times 50 + 20$ (parameter 2 of the 68020) = 220 ns to complete the access, allowing for a timing margin of $220 - 218 = 2 \text{ ns}$. Therefore, the State PAL must be a 22V10-15. The consecutive accesses within the same page require:

S0, S1 Cycles	50 ns
$\overline{\text{DS}}$ to CAS _{IEN}	10 ns
CAS _{IEN} to $\overline{\text{CAS}}_n$	17 ns
DRAM Page Mode Access	30 ns
Buffer Delay	7 ns
Data Setup	5 ns
Total Access Time	119 ns

With a total access time of 119 ns, the access is completed in 2.5 cycles with a margin of $(2 \times 50 + 20) - 119 = 1 \text{ ns}$ minimum. Since this is a worst-case timing analysis, this margin is sufficient.

The Timer PAL can be a 20X10A PAL; the only requirement is that it operates at a frequency of 20 MHz. For the 20X10A, the maximum frequency is 22 MHz. For the Byte Decoder PAL, the important parameter is $\overline{\text{BE}}[1:0]$, which must be valid soon enough for the $\overline{\text{CAS}}$ outputs to guarantee the data hold time of the DRAMs. Data Setup $\overline{\text{DS}}$ is 38 ns minimum and the data is held on the bus for 10 ns minimum after $\overline{\text{DS}}$ is deasserted. The data hold time required for the DRAMs is 20 ns. The propagation delay from $\overline{\text{CAS}}_n$ to $\overline{\text{CAS}}$ is 17 ns for a load of 112 pF. This leaves 11 ns from $\overline{\text{DS}}$ to $\overline{\text{BE}}_n$ ($38 + 10 - 20 - 17 = 11$). Therefore, a D-speed PAL is needed for the Byte Decoder. Figure 6 shows the timing for the byte-enable signals.

The $\overline{\text{RAS}}$ precharge time for 120 ns DRAMs is 90 ns, therefore two processor cycles ($50 \times 2 = 100 \text{ ns}$) are sufficient.



Table 2 shows the number of cycles needed to access memory for different processor and memory speeds for this architecture. The table shows that at 25 MHz, even the fastest DRAMs, 70 ns access times, require 1 wait state when accessed in the fast-page mode. Therefore, a different memory architecture would be needed to obtain higher memory and system performance at this processor speed.

Table 2. Access Cycles for Different Processor and Memory Speeds

Memory Speed (ns)	Processor Speed-ns					
	16 MHz		20 MHz		25 MHz	
	Initial	Cache Mode	Initial	Cache Mode	Initial	Cache Mode
120	5	3	5	3	6	4
100	5	3	5	3	6	4
80	4	3	5	3	5	4
70	4	3	4	3	5	4

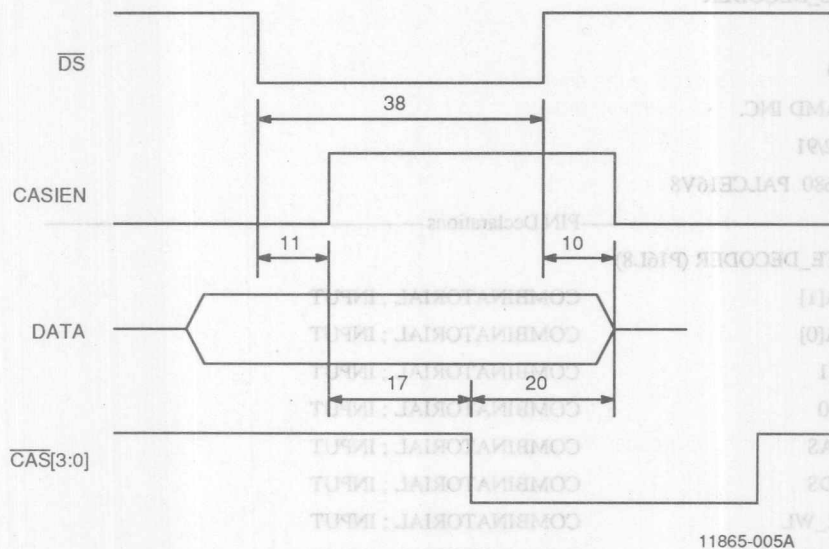


Figure 6. Byte Enable Timing Diagram

PARTS LIST

Part	Count
PALCE16V8	1
PALCE22V10	1
PAL20X10A	1
Am29C668	1
Decoder	1
74F245	4
Memories	64
Total	73

PAL Equations

The following application notes are guides to interfacing the Am29C668 with popular microprocessors. They are paper designs only, and have not been built and tested.

The following are the logic equations for the three PAL devices. They are written in PALASM.

Byte Decoder PAL. This PAL generates the byte enable signals for the Am29C668. It will also generate the Write Enable Signals to the DRAMs and the CASIEN signal to the Am29C668.

Note: BE[0] is the same as UUD, BE[1] = UMD, BE[2] = LMD, BE[3] = LLD in Motorola notation."

;PALASM Design Description

;-----Declarations Segment-----

TITLE BYTE_DECODER

PATTERN A

REVISION 1.0

COMPANY AMD INC.

DATE 07/12/91

CHIP _B_DE680 PALCE16V8

;-----PIN Declarations-----

DEVICE BYTE_DECODER (P16L8)

PIN 1	A[1]	COMBINATORIAL ; INPUT
PIN 2	A[0]	COMBINATORIAL ; INPUT
PIN 3	S1	COMBINATORIAL ; INPUT
PIN 4	S0	COMBINATORIAL ; INPUT
PIN 5	/AS	COMBINATORIAL ; INPUT
PIN 6	/DS	COMBINATORIAL ; INPUT
PIN 8	R_WL	COMBINATORIAL ; INPUT
PIN 9	/MEMREQ	COMBINATORIAL ; INPUT
PIN 11	PROGRAM	COMBINATORIAL ; INPUT
PIN 12	CASIEN	COMBINATORIAL ; OUTPUT
PIN 13	RL	COMBINATORIAL ; OUTPUT
PIN 14	/BE[0]	COMBINATORIAL ; OUTPUT
PIN 15	/BE[1]	COMBINATORIAL ; OUTPUT
PIN 16	/BE[2]	COMBINATORIAL ; OUTPUT
PIN 17	/BE[3]	COMBINATORIAL ; OUTPUT
PIN 18	/WE[0]	COMBINATORIAL ; OUTPUT
PIN 19	/WE[1]	COMBINATORIAL ; OUTPUT

;

;-----Boolean Equation Segment-----

EQUATIONS

BE[0] = /A[1] * /A[0]

BE[1] = /A[1] * A[0] + /A[1] * /S0 + /A[1] * S1

BE[2] = A[1] * /A[0] + /A[1] * /S1 * /S0 + /A[1] * S1 * S0 + /A[1] * A[0] * S1

BE[3] = /S1 * /S0 + A[1] * A[0] + A[1] * S1 + S1 * S0 * A[0]


```

RL = PROGRAM * AS
WE[1] = R_WL
WE[0] = R_WL
CASIEN = MEMREQ * DS

```

Simulation Segment

SIMULATION

SIMULATION

DATE 07/12/81
CHIP TIME 680 PAL 20X10

Timer PAL for Am29C668 to 68020 Interface. This PAL will generate refresh request for the DRAM controller."

;PALASM Design Description

----- Declaration Segment -----

TITLE TIMER

PATTERN A

REVISION 1.0

COMPANY AMD INC.

DATE 07/12/91

CHIP _TIMER680 PAL20X10

----- PIN Declarations -----

PIN 1	CLOCK	COMBINATORIAL ; INPUT
PIN 2	RESET	COMBINATORIAL ; INPUT
PIN 3	/RFRQ	COMBINATORIAL ; INPUT
PIN 13	/OE	COMBINATORIAL ; INPUT
PIN 14	/REFRESH	REGISTERED ; OUTPUT
PIN 15	/RCT[0]	REGISTERED ; OUTPUT
PIN 16	/RCT[1]	REGISTERED ; OUTPUT
PIN 17	/RCT[2]	REGISTERED ; OUTPUT
PIN 18	/RCT[3]	REGISTERED ; OUTPUT
PIN 19	/RCT[4]	REGISTERED ; OUTPUT
PIN 20	/RCT[5]	REGISTERED ; OUTPUT
PIN 21	/RCT[6]	REGISTERED ; OUTPUT
PIN 22	/RCT[7]	REGISTERED ; OUTPUT
PIN 23	/RCT[8]	REGISTERED ; OUTPUT
PIN 12	GND	; INPUT
PIN 24	VCC	; INPUT

;SPECIAL DEFINITIONS

STRING START_REFRESH '/RCT[8] * /RCT[7] * /RCT[6] * /RCT[5] * /RCT[4] * /RCT[3] * /RCT[2] * /RCT[1] * /RCT[0]'

STRING INIT '/RCT[8] * /RCT[7] * /RCT[6] * /RCT[5] * /RCT[4] * /RCT[3] * /RCT[2] * /RCT[1] * /RCT[0]'

----- Boolean Equation Segment -----

EQUATIONS

REFRESH := START_REFRESH * /RESET + REFRESH * /RESET * /RFRQ

RCT[0] := (/RCT[0]) :+ (INIT)

RCT[1] := (/RCT[0]) :+ (RCT[1] + INIT)

RCT[2] := (/RCT[1] * /RCT[0]) :+ (RCT[2] + INIT)

RCT[3] := (/RCT[2] * /RCT[1] * /RCT[0]) :+ (RCT[3] + INIT)

RCT[4] := (/RCT[3] * /RCT[2] * /RCT[1] * /RCT[0] + INIT) :+ (RCT[4])

RCT[5] := (/RCT[4] * /RCT[3] * /RCT[2] * /RCT[1] * /RCT[0] + INIT) :+ (RCT[5])

RCT[6] := (/RCT[5] * /RCT[4] * /RCT[3] * /RCT[2] * /RCT[1] * /RCT[0]) :+ (RCT[6] + INIT)

RCT[7] := (/RCT[6] * /RCT[5] * /RCT[4] * /RCT[3] * /RCT[2] * /RCT[1] * /RCT[0]) :+ (RCT[7] + INIT)

RCT[8] := (/RCT[7] * /RCT[6] * /RCT[5] * /RCT[4] * /RCT[3] * /RCT[2] * /RCT[1] * /RCT[0]) ;+ (RCT[8])

Simulation Segment

SIMULATION

PALASM Design Description		
Description Segment		
TITLE STATE		
PATTERN A		
REVISION 1.0		
COMPANY AMD INC.		
DATE 07/12/91		
CHIP_STATE680 PAL22V10		
PIN Definitions		
PIN 1	CLOCK	COMBINATORIAL ; INPUT
PIN 2	CLK	COMBINATORIAL ; INPUT
PIN 3	/AS	COMBINATORIAL ; INPUT
PIN 4	/DS	COMBINATORIAL ; INPUT
PIN 5	/MEMREQ	COMBINATORIAL ; INPUT
PIN 6	/REFRESH	COMBINATORIAL ; INPUT
PIN 7	PROGRAM	COMBINATORIAL ; INPUT
PIN 8	RESET	COMBINATORIAL ; INPUT
PIN 9	/CH	COMBINATORIAL ; INPUT
PIN 12	RAS	COMBINATORIAL ; OUTPUT
PIN 16	/DSACK[0]	COMBINATORIAL ; OUTPUT
PIN 17	/DSACK[1]	COMBINATORIAL ; OUTPUT
PIN 22	/OHT	COMBINATORIAL ; OUTPUT
PIN 23	/DSACKEN	COMBINATORIAL ; OUTPUT
PIN 19	/MSTATE[0]	REGISTERED ; OUTPUT
PIN 20	/MSTATE[1]	REGISTERED ; OUTPUT
PIN 21	/MSTATE[2]	REGISTERED ; OUTPUT
PIN 18	/RRO	REGISTERED ; OUTPUT
PIN 13	GND	INPUT
PIN 24	VCC	INPUT
SPECIAL DEFINITIONS		
STRING IDLE_ST 'B000'		
STRING SW1_ST 'B001'		
STRING SW2_ST 'B011'		
STRING ACC_ST 'B111'		
STRING PM_ST 'B101'		
STRING FCI_ST 'B110'		
STRING FCS_ST 'B010'		
STRING FCI2_ST 'B100'		

State PAL for Am29C668 to 68020 Interface. This PAL contains the state machine that controls the memory. It will arbitrate between refresh and memory accesses and will generate the control signals for the Am29C668 and 68020 processor."

;PALASM Design Description

;----- Declaration Segment -----

TITLE STATE

PATTERN A

REVISION 1.0

COMPANY AMD INC.

DATE 07/12/91

CHIP _STATE680 PAL22V10

;----- PIN Declarations -----

PIN 1	CLOCK	COMBINATORIAL ; INPUT
PIN 2	CLK	COMBINATORIAL ; INPUT
PIN 3	/AS	COMBINATORIAL ; INPUT
PIN 4	/DS	COMBINATORIAL ; INPUT
PIN 5	/MEMREQ	COMBINATORIAL ; INPUT
PIN 6	/REFRESH	COMBINATORIAL ; INPUT
PIN 7	PROGRAM	COMBINATORIAL ; INPUT
PIN 8	RESET	COMBINATORIAL ; INPUT
PIN 9	/CH	COMBINATORIAL ; INPUT
PIN 15	RASI	COMBINATORIAL ; OUTPUT
PIN 16	/DSACK[0]	COMBINATORIAL ; OUTPUT
PIN 17	/DSACK[1]	COMBINATORIAL ; OUTPUT
PIN 22	/PGHIT	COMBINATORIAL ; OUTPUT
PIN 23	/DSACKEN	COMBINATORIAL ; OUTPUT
PIN 19	MSTATE[0]	REGISTERED ; OUTPUT
PIN 20	MSTATE[1]	REGISTERED ; OUTPUT
PIN 21	MSTATE[2]	REGISTERED ; OUTPUT
PIN 18	/RFRQ	REGISTERED ; OUTPUT
PIN 12	GND	; INPUT
PIN 24	VCC	; INPUT

;SPECIAL DEFINITIONS

STRING IDLE_ST '#B000'

STRING SW1_ST '#B001'

STRING SW2_ST '#B011'

STRING ACC_ST '#B111'

STRING PM_ST '#B101'

STRING PC1_ST '#B110'

STRING PC2_ST '#B010'

STRING PC12_ST '#B100'

```

STRING IDLE '/MSTATE[2] * /MSTATE[1] * /MSTATE[0]'
STRING SW1 '/MSTATE[2] * /MSTATE[1] * MSTATE[0]'
STRING SW2 '/MSTATE[2] * MSTATE[1] * MSTATE[0]'
STRING ACC 'MSTATE[2] * MSTATE[1] * MSTATE[0]'
STRING PM 'MSTATE[2] * /MSTATE[1] * MSTATE[0]'
STRING PC1 'MSTATE[2] * MSTATE[1] * /MSTATE[0]'
STRING PC12 'MSTATE[2] * /MSTATE[1] * /MSTATE[0]'
STRING PC2 '/MSTATE[2] * MSTATE[1] * /MSTATE[0]'

```

_____ Boolean Equation Segment _____

EQUATIONS

```

RASI = IDLE * RFRQ * /CLK + IDLE * AS * MEMREQ * CLK * /REFRESH + IDLE * RASI + SW1 + SW2 + ACC
      + PM * /(CH * AS * MEMREQ * CLK) * RASI

```

```

DSACKEN = (MEMREQ + PROGRAM) * DS

```

```

PGHIT = PM * AS * MEMREQ * CH * CLK * RASI + PM * PGHIT * /CLK

```

```

RFRQ = /RESET * REFRESH * /PROGRAM * (IDLE * /RASI + PC1 + PC12) + RFRQ * /RESET * /PC1

```

```

DSACK[1] = SW2 * /RFRQ * /CLK + ACC * /RFRQ + PM * /CLK * PGHIT + PROGRAM * AS

```

```

DSACK[0] = SW2 * /RFRQ * /CLK + ACC * /RFRQ + PM * /CLK * PGHIT + PROGRAM * AS

```

```

IF (RESET) THEN

```

```

    BEGIN

```

```

        MSTATE[2..0] := IDLE_ST

```

```

        RFRQ := 0

```

```

    END

```

```

CASE (MSTATE[2..0])

```

```

    BEGIN

```

```

        IDLE_ST:

```

```

            BEGIN

```

```

                IF (RASI) THEN

```

```

                    BEGIN

```

```

                        MSTATE[2..0] := SW1_ST

```

```

                    END

```

```

                ELSE

```

```

                    BEGIN

```

```

                        MSTATE[2..0] := IDLE_ST

```

```

                    END

```

```

            END

```

```

        SW1_ST:

```

```

            BEGIN

```

```

                MSTATE[2..0] := SW2_ST

```

```

            END

```

```

        SW2_ST:

```

```

            BEGIN

```



```

MSTATE[2..0] := ACC_ST
END
ACC_ST:
BEGIN
  IF (REFRESH + RFRQ) THEN
    BEGIN
      MSTATE[2..0] := PC1_ST
    END
  ELSE
    BEGIN
      MSTATE[2..0] := PM_ST
    END
  END
PM_ST:
BEGIN
  IF (PGHIT) THEN
    BEGIN
      MSTATE[2..0] := ACC_ST
    END
  ELSE
    BEGIN
      IF (/RASI) THEN
        BEGIN
          MSTATE[2..0] := PC12_ST
        END
      ELSE
        BEGIN
          IF (REFRESH) THEN
            BEGIN
              MSTATE[2..0] := PC12_ST
            END
          ELSE
            BEGIN
              MSTATE[2..0] := PM_ST
            END
          END
        END
      END
    END
  END
PC1_ST:
BEGIN
  MSTATE[2..0] := PC2_ST

```

```
END
PC12_ST:
BEGIN
  IF (REFRESH) THEN
    BEGIN
      MSTATE[2..0] := PC2_ST
    END
  ELSE
    BEGIN
      MSTATE[2..0] := IDLE_ST
    END
  END
END
PC2_ST:
BEGIN
  IF (RFRQ) THEN
    BEGIN
      MSTATE[2..0] := SW1_ST
    END
  ELSE
    BEGIN
      MSTATE[2..0] := IDLE_ST
    END
  END
END
END ;"CASE"
;----- Simulation Segment -----
SIMULATION
;
```

```

END
PC12_ST
BEGIN
    IF (REFRESH) THEN
        BEGIN
            MSTATE[2:0] := PC2_ST
        END
    ELSE
        BEGIN
            MSTATE[2:0] := IDLE_ST
        END
    END
END
PC3_ST
BEGIN
    IF (REQ) THEN
        BEGIN
            MSTATE[2:0] := SW1_ST
        END
    ELSE
        BEGIN
            MSTATE[2:0] := IDLE_ST
        END
    END
END
END ; "CASE"

```

Simulation segment

SIMULATION

Am29C668 Configurable Dynamic Memory Controller to 80486 Microprocessor



INTRODUCTION

The interface between the Am29C668 4-Mbit Configurable Dynamic Memory Controller (CDMC) and the 80486 microprocessor was designed for maximum performance, while using relatively inexpensive dynamic RAMs. This design uses 80-ns fast-page-mode DRAMs, yet achieves single-cycle burst accesses at 25 MHz. The memory system is intended to be as general as possible so that users may tailor their implementations to specific memory systems. A block diagram, timing analyses and logic equations necessary to implement the design are included.

Distinctive Characteristics

- Am29C668 4-Mbit Configurable Dynamic Memory Controller/Driver
- 25-MHz 80486 Microprocessor
- 80-ns Fast-Page-Mode 1 Mbit x 9 DRAMs Single In-line Memory Modules (SIMMs). Supports up to 32 Mbytes using 4-Mbit DRAMs, 8 Mbytes with 1-Mbit DRAMs.
- Uses Common Data Input and Output; Always Uses Write-Early Cycles.
- Two Interleaved Banks of 32-Bit Memory
- Four-Cycle Initial Access, Single-Cycle Burst Accesses for Read Cycles. Three-Cycle Access Within a Page.

80486 Overview

Dynamic RAMs can provide far more memory at lower cost and power in the available board space than is possible when using static RAMs (SRAMs). The main penalty in using DRAMs is a loss of speed in the initial memory-access time. Burst-access performance can be maintained by the use of bank interleaving and fast-page-mode DRAMs. The 80486 has several architectural features to minimize the latency of DRAM accesses.

The 80486 has an on-chip 8-Kbyte cache that is four-way set associative with a pseudo least recently used (LRU) replacement algorithm. The set size is 2 Kbyte with a line size of 16 bytes, yielding 128 lines per set. Only full lines (16 bytes) can be valid; there is no provision for partially filled lines. The external hardware can determine which memory locations can be cached by

asserting the Cache-Enable \overline{KEN} pin one cycle before a memory Read is completed. Software can also be used to control the cache. The 80486 supports only Write-through, not Write-back. A miss during a Write does not result in a cache allocation. This means that Write cycles are always single memory accesses, never burst accesses.

The initial Read and Write cycles are similar. During the first cycle, the 80486 asserts the address, byte enables, and control signals. The address status output \overline{ADS} is asserted, signaling the external memory that the address and control are valid. In the second cycle, the external memory drives the data on the bus for Reads, or writes the data to the memory during Writes. The ready signal \overline{RDY} is asserted to indicate that the memory access is complete. If the memory needs more cycles to complete the access, it does not assert \overline{RDY} until the cycle when the access completes.

During a Read to an external memory address that can be cached, the external memory asserts \overline{KEN} . The 80486 tries to read the whole line into the cache. The first word read is always the current word needed for the instruction execution. The 80486 will not assert the Burst Last signal \overline{BLAST} to signify that the processor wants a burst access. If the memory can support burst accesses, it asserts Burst Ready \overline{BRDY} instead of Non-Burst Ready \overline{RDY} when the access is complete. The next three words are determined by the address of the initial word. Table 1 shows the order of the words accessed during a cache fill.

The cache can be filled using burst cycles or interrupted burst cycles. The initial access always requires at least two cycles. For burst cycles, the processor can accept data every clock cycle. If the memory cannot access data at this rate, wait states can be inserted by not asserting \overline{BRDY} . This memory design uses single-cycle burst accesses to fill the cache. Interrupted burst cycles occur when the external memory asserts \overline{RDY} when the

Table 1. Cache Fills

First Address	Second Address	Third Address	Fourth Address
0	4	8	C
4	0	C	8
8	C	0	4
C	8	4	0

80486 requests burst cycles. The 80486 then uses the initial access to read the next cache location. This greatly reduces the available memory bandwidth and should be avoided. This design does not use interrupted burst cycles. All Write cycles are limited to one word (32-bits) and are at least two cycles.

The 80486 supports bus byte parity. It generates the parity bits during data Writes and checks parity during data Reads. If a parity error occurs, the 80486 asserts Parity Status output PCHK. The system must use the PCHK signal to generate an interrupt, since the 80486 does nothing on a parity error other than drive PCHK active. If parity is not used, the Data Parity pins DP[0-3] should be connected to Vcc through a pull-up resistor.

This design implements parity, which only detects single-bit errors in the memory; it does not correct them. A better method to insure memory reliability is to use the Am29C660 Error Detection and Correction (EDC) circuit. This device contains the logic necessary to generate check bits on a 32-bit data field according to a modified Hamming-code algorithm. It can also correct single-bit errors in the data word when check bits are provided. The EDC circuit detects all single and double-bit errors as well as some triple-bit errors. For 32-bit words, seven check bits are used. This is the preferable method to use when building large, highly reliable memories.

MEMORY ARCHITECTURE OVERVIEW

To obtain maximum memory throughput, but still maintain a reasonable cost, 80-ns fast-page-mode 1-Mbit DRAMs are used. For this memory design, the 25-MHz 80486 completes the initial access to memory in four cycles. If the access is to a page that is currently active, the access is completed in three cycles.

Fast-page-mode DRAMs appear to the processor as if they are fast cache memories during accesses within the page. The page size for a 1-Mbit DRAM is 1024 bits or 1 Kbits. This memory is 32 bits wide and two banks are interleaved, therefore its page size is 8 Kbytes. The Am29C668 detects accesses within the same page via the on-chip cache-mode operation. When a new address is latched, it is compared with the previous row and bank address. If the addresses are the same, \overline{CH} is asserted. The memory state machine immediately begins the next access. An access outside the page, a page miss, causes the memory controller to perform the RAS precharge for the DRAMs followed by a normal memory access. The total access time on a page miss requires six cycles, one for decoding, two cycles for the RAS precharge and three for the data access. Shorter memory-access times result when using the cache-mode method than when using normal DRAM ac-

cesses. The actual performance of the memory system depends upon the instruction mix of the programs executed.

Each Am29C668 controls a memory array consisting of one bank, containing four 1 Mbit x 9 SIMMs or 32 bits of memory plus 4 bits for byte parity. This provides 4 Mbytes of memory in each bank, totaling 8 Mbytes for the design. For larger memories, 4 Mbit x 9 SIMMs are used; for smaller memories 256 Kbits x 9 SIMMs can be used.

Refresh Cycles

To retain data, dynamic memories must be refreshed periodically to restore the charge on the memory-cell storage capacitors. For 1-Mbit DRAMs, all 512 rows of memory must be refreshed every 8 ms. There are three different methods to perform the refresh cycles: burst, forced and hidden; each has its advantages and disadvantages. The best method is determined by the instruction mix, system hardware and performance requirements.

The burst-refresh method refreshes all 512 rows sequentially and works especially well in systems with long idle times between memory accesses. The main disadvantage of this system is an access to memory may be delayed for long periods during the refresh cycles, greatly impacting system response time. This would definitely not be an acceptable method for real-time systems.

The forced or distributed-refresh method periodically insert refresh cycles. If refreshes are interspersed between memory accesses, the memory-access time is not greatly impacted, since there is a low probability of refresh-request and memory-request contention. One refresh request is generated every $15.6 \mu s = 8 \text{ ms}/512$ rows. This method is preferable to burst refresh in most systems.

Even better would be for the refresh cycles to occur when the processor was accessing other memory or I/O. This type of refreshing is called hidden refresh since all or most of the refresh cycle is overlapped with another access. There are times however, when the system continually accesses the same memory and does not permit hidden refreshes to be performed. If this happens, a forced refresh cycle must be performed. Hidden refresh has the lowest system impact since all or most of the refresh cycle is overlapped with an access to another memory or I/O device. There are conceivable situations where hidden refresh would not perform as well as forced refresh; however, for most general applications, hidden refresh is the best choice.

For several reasons, this design utilizes forced refreshes instead of hidden refreshes. First, to obtain maximum performance using page-mode accesses, the design must maximize the number of accesses to the active page. This minimizes the total number of cycles by amortizing the slower initial access over the faster page-mode accesses. A hidden-refresh cycle will terminate the page-mode access, diminishing the performance gained by using the page-mode access. Secondly, additional logic is needed to keep track of when hidden-refresh cycles are performed. This logic must suppress the forced-refresh request after a hidden-refresh cycle is performed and must force a refresh when no hidden refresh is performed. This adds extra devices and consumes more board space, money and power.

The external control logic asserts RAS \bar{S} and CAS \bar{S} to generate the refresh timing and the Am29C668 asserts \overline{CAS} before \overline{RAS} . In this refresh mode (MC[1:0] = 00), the RAS \bar{S} input controls the CAS \bar{S} outputs and the CAS \bar{S} input controls the \overline{RAS} outputs. Thus, external logic does not have to control RAS \bar{S} and CAS \bar{S} differently during refreshes. $\overline{REFRESH}$ is asserted by the 74F269 counter every 9.64 μ s to request a refresh access. If a memory access is in progress, the access is completed before the refresh cycle begins. Burst-memory accesses are never interrupted by refresh requests. The processor can continue to execute out of the internal cache while the memory is being refreshed; this keeps refresh overhead to a minimum. If both a memory access request and refresh request occur during the same cycle, the refresh request is given priority to insure that the refresh requirements of the DRAM are met.

FUNCTIONAL DESCRIPTION

Figure 1 is the main block diagram for this design. Figure 2A shows a simple initial memory access, Figure 2B is a burst access followed by a cache hit, Figure 2C shows a cache miss and Figure 2D shows a programming cycle.

Am29C668 CDMC

The Am29C668 generates the \overline{RAS} , \overline{CAS} and address signals to the DRAM array. No external drivers are needed.

The Am29C668 must be programmed via an I/O access before any memory accesses may occur. The actual decoding is left up to the user since it is system dependent. The address bits A[12:3] contain the value to be loaded into the configuration register. For this design, the lower 13 address bits are 00X11XX1110XXXX (where X is don't care). The options selected are: External Timing, Cache Mode, \overline{CAS} before \overline{RAS} Refresh, Byte \overline{CAS} Decoding and 2-Bank Memory Configuration. Note: AC[10] is only used with 4-Mbit DRAMs; in applications using only 1-Mbit DRAMs, AC[10] can be tied Low.

The Am29C668 supports byte decoding through its \overline{CASEN} [3:0] inputs. Read accesses always access all 32 bits. The Byte Enable signals \overline{BE} [3:0] from the 80486 determines which bytes will be written to memory. When \overline{BE} [0] is asserted, Data Lines D[7:0] are written to memory; when \overline{BE} [1] is asserted, D[15:8] are written to memory; when \overline{BE} [2] is asserted, D[23:16] are written to memory; when \overline{BE} [3] is asserted, D[31:24] are written to memory. The Control PAL device generates the \overline{CASEN} [3:0] signals for the Am29C668 by decoding the Write/Read W/R signal and the \overline{BE} signals from the 80486.

The Address Latch Enable ALE is tied High; therefore, the Am29C668 address latches are transparent. This is necessary so that the external logic can generate the next column address during burst-mode accesses. The 80486 holds all the bus control signals valid until after the memory access is complete, therefore the lower address bits do not have to be latched. When ALE is High, the address latch of the Am29C668 is transparent. When ALE is Low, the address is latched.

The Register Load/Column Counter (RL/CC) input of the Am29C668 is a dual function input. In the initialize mode (MC[1:0] = 11), this input is the register load signal. On the rising edge of RL/CC, the inputs at AC[10:0] are stored in the configuration register. The counter function is not used in this design; only the RL function is used to load the configuration register. The external control logic generates one signal to control MSEL and RL/CC. This was done to minimize the number of external devices needed. MSEL controls the multiplexing of the address to the DRAMs. When MSEL is Low, the row address is output; when High, the column address is output. MSEL goes from High to Low only during \overline{RAS} refresh, so tying MSEL to RL/CC causes no problems.

The Am29C668 has on-chip comparators for implementing cache-mode operation. When properly configured, the Am29C668 compares the previously accessed row and bank address with the current row and bank address. If they are the same, the Cache Hit (\overline{CH}) output is asserted. The memory-control logic can then access the memory in a fast-page or static-column mode. In this design, cache-mode accesses require three cycles instead of four cycles for initial accesses. If the access is to a different row, the control logic must insert a \overline{RAS} precharge and then perform a normal access. This takes six cycles to complete. Bank Select inputs SEL[1:0] must be tied Low to insure proper comparison of the bank address. Row Address input AR[10] is only used with 4-Mbit DRAMs. To insure proper operation of the cache mode with 1-Mbit DRAMs, AR[10] must be tied Low during memory accesses.

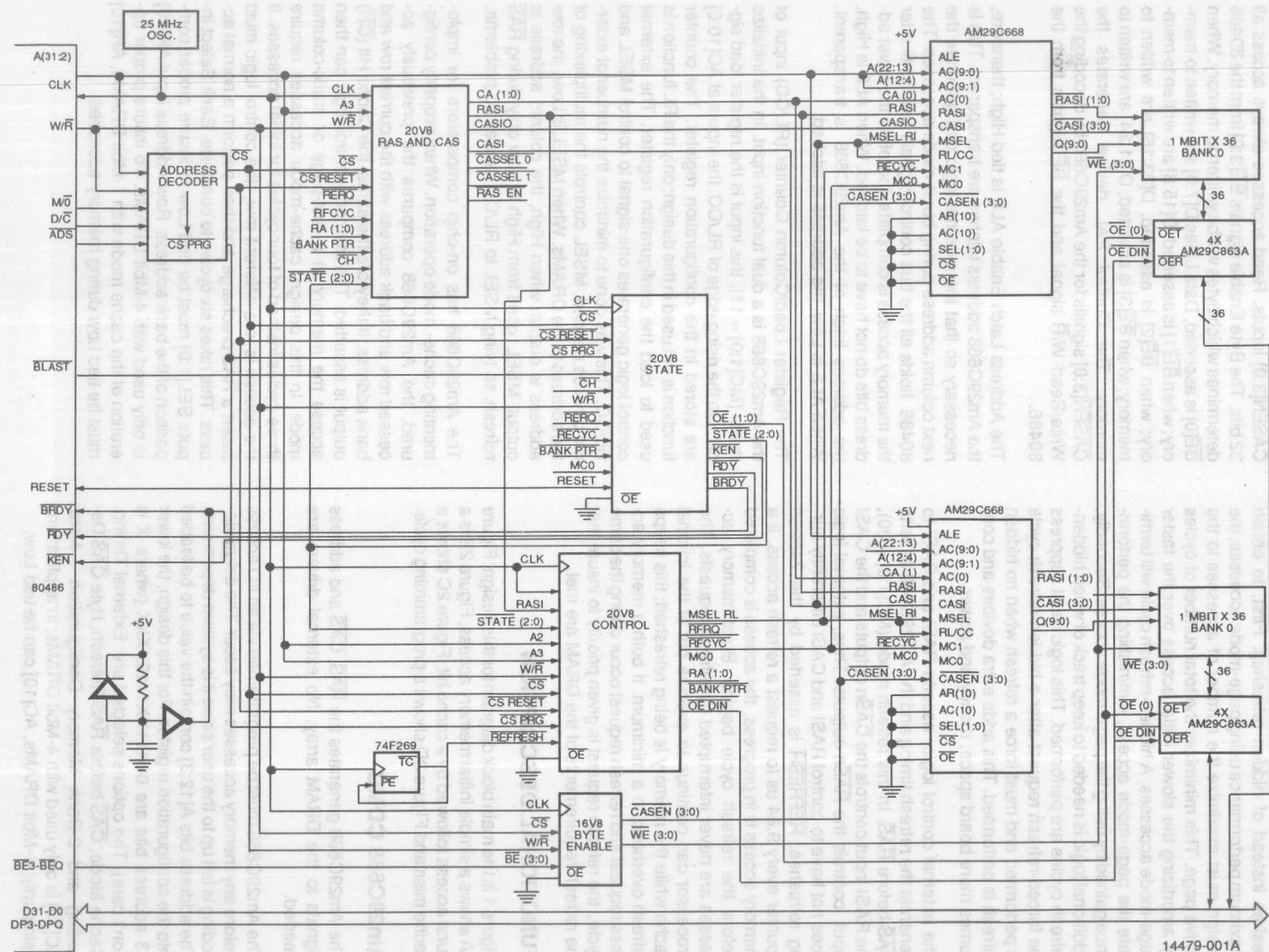


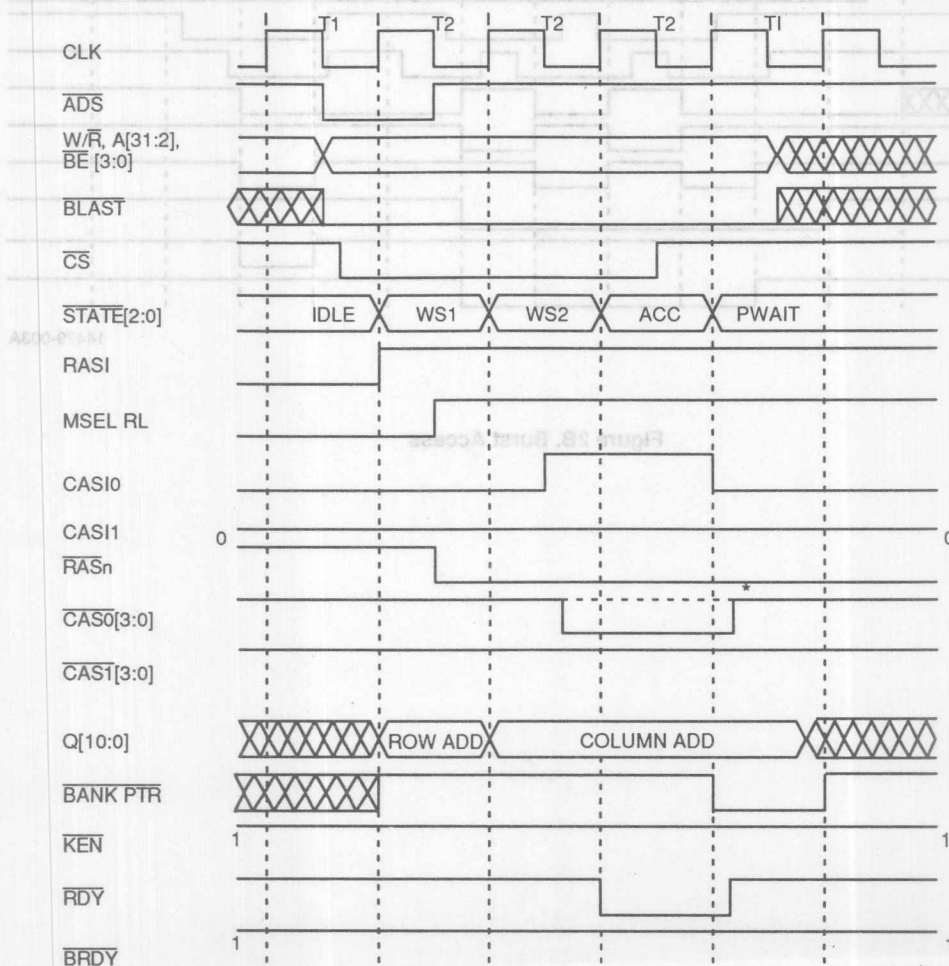
Figure 1. Interface Block Diagram

14479-001A

This design uses $\overline{\text{CAS}}$ -before- $\overline{\text{RAS}}$ refresh cycles, which are similar to the initial memory access except that $\overline{\text{CAS}}$ is asserted to the DRAMs before $\overline{\text{RAS}}$. When the DRAMs detect this refresh cycle, they use internal row-refresh counters instead of the external address. This feature is available on most DRAMs 1 Mbit and larger and can reduce the refresh overhead by eliminating the time needed for the Am29C668 to drive the row-refresh address to the DRAMs. The Am29C668 supports this refreshing mode by internally multiplexing the $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$ signals. The external control logic does not have to generate $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$ in a different order. During refresh cycles, $\overline{\text{RAS}}$ controls the $\overline{\text{CASn}}$ outputs and $\overline{\text{CAS}}$ controls the $\overline{\text{RASn}}$ outputs. This helps simplify the external control logic.

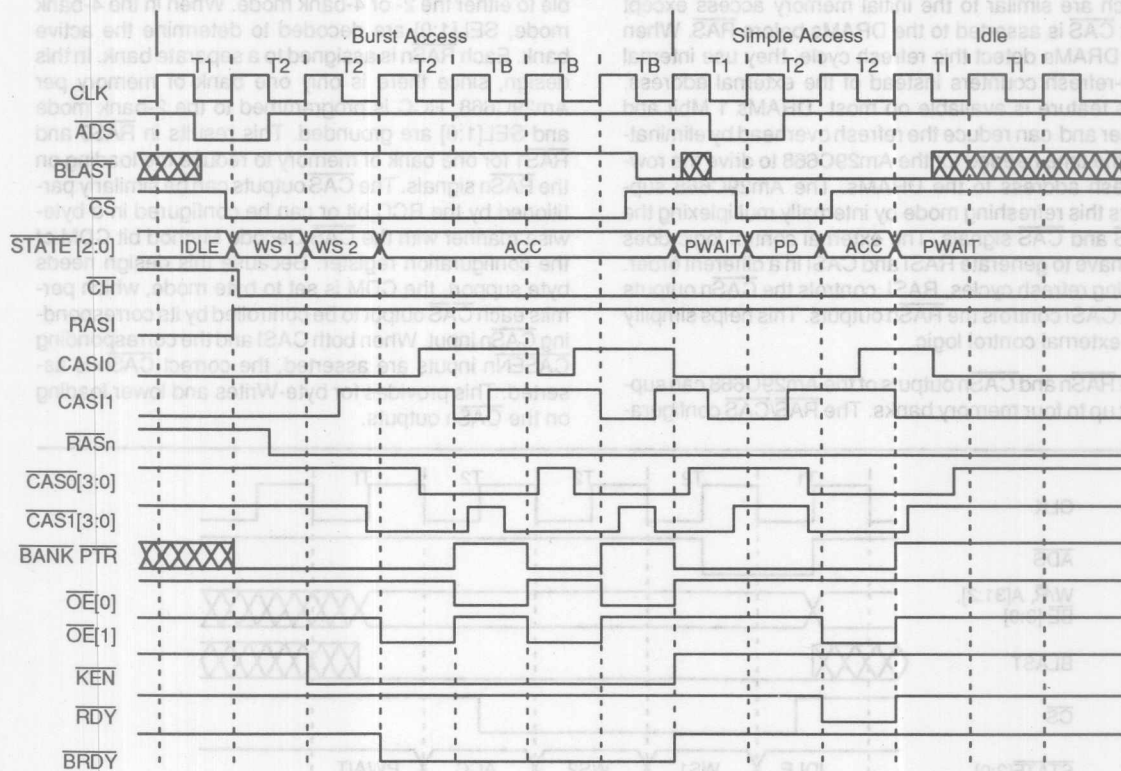
The $\overline{\text{RASn}}$ and $\overline{\text{CASn}}$ outputs of the Am29C668 can support up to four memory banks. The $\overline{\text{RAS/CAS}}$ configura-

tion bit RCC of the configuration register is programmable to either the 2- or 4-bank mode. When in the 4-bank mode, $\text{SEL}[1:0]$ are decoded to determine the active bank. Each $\overline{\text{RASn}}$ is assigned to a separate bank. In this design, since there is only one bank of memory per Am29C668, RCC is programmed to the 2-bank mode and $\text{SEL}[1:0]$ are grounded. This results in $\overline{\text{RAS0}}$ and $\overline{\text{RAS1}}$ for one bank of memory to reduce the loading on the $\overline{\text{RASn}}$ signals. The $\overline{\text{CAS}}$ outputs can be similarly partitioned by the RCC bit or can be configured in a byte-wise manner with the $\overline{\text{CAS}}$ Decode Method bit CDM of the configuration register. Because this design needs byte support, the CDM is set to byte mode, which permits each $\overline{\text{CAS}}$ output to be controlled by its corresponding $\overline{\text{CASn}}$ input. When both $\overline{\text{CAS}}$ and the corresponding $\overline{\text{CASn}}$ inputs are asserted, the correct $\overline{\text{CASn}}$ is asserted. This provides for byte-Writes and lower loading on the $\overline{\text{CASn}}$ outputs.



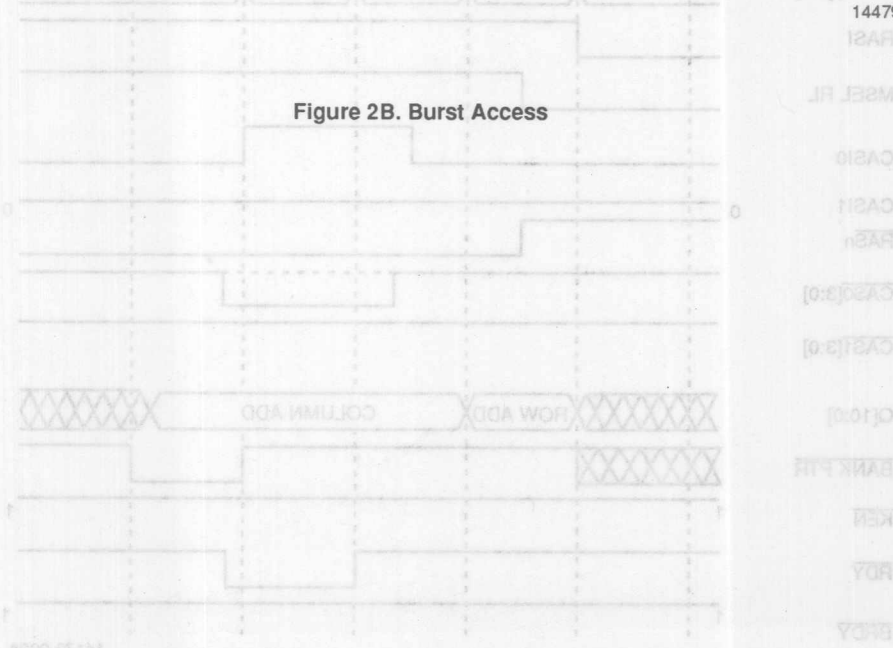
*All $\text{CAS0}[3:0]$ Low During Reads. $\text{CAS0}[3:0]$ follows BE0-BE3 during Writes.

Figure 2A. Simple Initial Access



14479-003A

Figure 2B. Burst Access



* All CAS0[3:0] Low During Reads. CAS0[3:0] follows BE0-BE3 during Writes.

Figure 2A. Simple Initial Access

Address Decoder

An address decoder is needed to distinguish between the different 80486 memory and I/O requests. Different systems will want to map the memory and the Am29C668 configuration register to different locations. The configuration register can either be memory mapped or an output port. The only requirement of the address decoder in this design is that it must latch the chip selects until $\overline{\text{RDY}}$ or $\overline{\text{BRDY}}$ is returned to the 80486. This design also implements a forced reset that the CPU can generate by asserting the input $\overline{\text{CS_RESET}}$ to the external logic. Again, this may be decoded as a memory or I/O access. Also, the CPU can assert the $\overline{\text{CS_PRG}}$ input to the external logic to program the Am29C668 configuration register. The following are possible examples of the chip select decodings:

$$\text{CS} = \text{MEM_ADDR} * \text{ADS} * \text{M_IO} + \text{CS} * (\text{RDY} + \text{BRDY});$$

$$\text{CS_PRG} = 668_ADDR * \text{ADS} * \text{M_IO} * \text{D_C} + \text{CS_PRG} * \text{RDY};$$

$$\text{CS_RESET} = \text{RESET_ADDR} * \text{ADS} * \text{M_IO} * \text{D_C} + \text{CS_PRG} * \text{RDY};$$

Where, MEM_ADDR is the memory address, 668_ADDR is the configuration-register address and RESET_ADDR is the software-reset address.

Byte-Enable PAL Device

This PAL generates the $\overline{\text{CAS}}$ byte enable to the Am29C668 and Write signals to the DRAMs. The following inputs are used:

CLK 25-MHz System Clock
 $\overline{\text{CS}}$ Chip Select from the Address Decoder
 W_R Write High, Read Low, from 80486
 $\overline{\text{BE}}[3:0]$ Byte Enables from 80486
 The following outputs are generated:

$\overline{\text{CASEN}}[3:0]$ $\overline{\text{CAS}}$ Enables to Am29C668s
 $\overline{\text{WE}}[3:0]$ Write Enables to DRAMs

The $\overline{\text{CASEN}}[3:0]$ outputs are connected to the Am29C668 $\overline{\text{CASEN}}[3:0]$ inputs, which control the $\overline{\text{CASn}}$ outputs of the Am29C668. All $\overline{\text{CASEN}}[3:0]$ are asserted during a Read cycle. During Write cycles, $\overline{\text{BE}}[3:0]$ from the 80486 determine which $\overline{\text{CASEN}}[3:0]$ outputs are to be asserted.

The $\overline{\text{WE}}[3:0]$ outputs are the Write Enable signals to the DRAMs. There is one output to each byte of memory to minimize the loading on these outputs; 33- Ω series resistors may be needed on these outputs to minimize the under-shoot during the High-to-Low transitions.

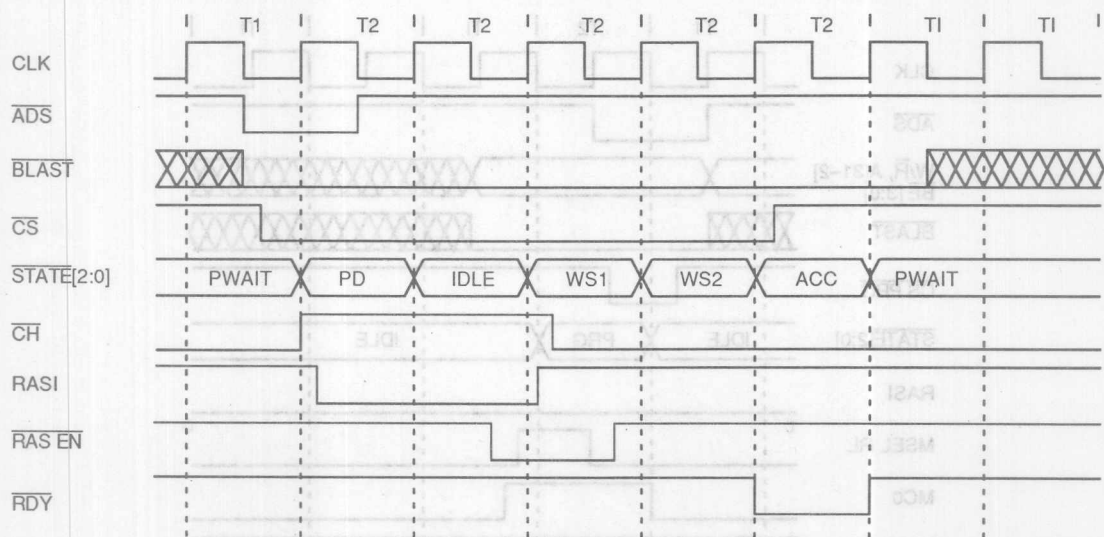


Figure 2C. Cache Miss

14479-004A

Control PAL Device

This PAL generates the refresh cycle signal for the Am29C668 and external control logic. The following inputs are used:

CLK	25-MHz System Clock, clocks the registers
CLK_COMB	25-MHz System Clock, used combinatorially
\overline{CS}	Chip Select, from the Address Decoder
$\overline{CS_RESET}$	Chip Select Reset, from the Address Decoder
$\overline{CS_PRG}$	Chip Select Program, from the Address Decoder
W_R	Write active High, Read active Low, from 80486
STATE[2:0]	State Variables, from State PAL device
REFRESH	Refresh Request, from Refresh Timer(74F269)
A2,A3	System Address 2 and 3
RASI	Row Address Strobe Input, from the RAS and CAS PAL

The following outputs are generated:

RFCYC	Refresh Cycle, to Am29C668, State PAL and RAS PAL devices
RFRQ	Refresh Request, connected to State PAL device to generate refresh cycles.
MC0	Mode control bit 0, to the Am29C668

MSEL_RL	Modes Select and Register Load, to the Am29C668
BANK_PTR	Bank Pointer, Indicates the currently active memory bank
RA[1:0]	Registered Address, to RAS and CAS PAL device
$\overline{OE_DIN}$	Output Enable Data In, Enables Write Data Drivers

The \overline{RFRQ} output is the refresh request signal to the state machine; it is asserted when the refresh counter is decremented to zero. \overline{RFRQ} remains asserted until the refresh cycle is completed; \overline{RFRQ} is asserted and the state is PD_ST. RFCYC is connected to the Am29C668 MC1 input. When RFCYC is asserted, the Am29C668 is in the refresh mode. RFCYC must be deasserted during programming and reset cycles.

MC0 is used to distinguish reset and program cycles from refresh and Read/Write cycles. During reset and program cycles, MC0 is asserted. During Read/Write and refresh cycles, MC0 is not asserted.

MSEL_RL is a dual function output, tied to the Am29C668 RL/CC and MSEL inputs. RL is only used during the program mode. When RL goes from Low to High and MC0 and MC1 are both High, the Am29C668 loads its configuration register with the value on the AC[10:0] inputs. In the normal Read/Write mode,

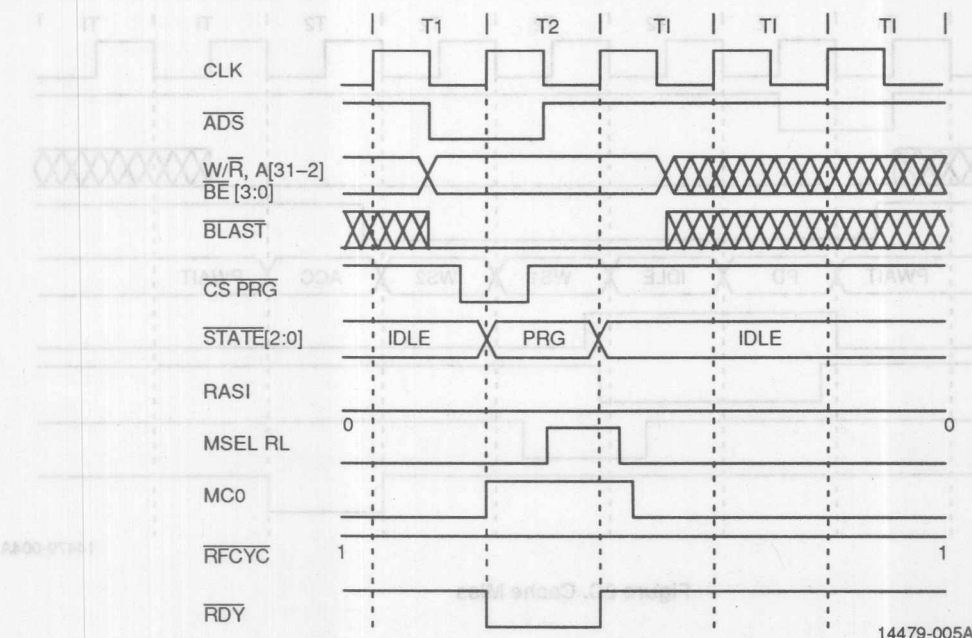


Figure 2D. Am29C668 Programming Cycle

RL/CC is the column counter. On the falling edge of RL/CC the internal column counter is incremented. In the normal Read/Write mode, MSEL is used to control the address multiplexer to the DRAMs. When MSEL is Low, the row address is output; when MSEL is High, the column address is High. There is no problem involved when tying RL to MSEL because ALE is tied High; therefore the address latch is transparent and RL/CC has no effect.

BANK_PTR is used to indicate the currently active bank. When **BANK_PTR** is asserted, bank 1 is active; otherwise bank 0 is active. **BANK_PTR** is negated during the **ACC_ST** to perform the bank interleaving. **BANK_PTR** is set by A3 during the **IDLE_ST** or **PD_ST**.

The **OE_DIN** put enables the data-output drivers used during Write cycles. These outputs are turned on in the second cycle of the memory access and remain on until the end of the memory access, to provide ample set-up and hold time to the DRAMs.

RA[1:0] are used in the burst counter that keeps track of the memory address. This is explained further in the RAS and CAS PAL.

RAS and CAS PAL Device

This PAL generates the Row Address Strobe Input (RASI), Column Address Strobe Input (CASI) and the lower address bit for the Am29C668s. The following inputs are used:

CLK	25 MHz System Clock
CS	Chip Select, from the Address Decoder
CS_RESET	Chip Select Reset, from the Address Decoder
RFRQ	Refresh Request, from Refresh Timer
RFCYC	Refresh Cycle, from Control PAL device
STATE[2:0]	State Variables, from State PAL device
CH	Cache Hit, from the Am29C668
W_R	Write active High, Read active Low, from 80486
RA[1:0]	Registered Address, from the State PAL device
A3	Address bit 3, from the 80486
BANK_PTR	Bank Pointer, from the State PAL device

The following outputs are generated:

RASI	Row Address Strobe Input, connected to Am29C668 RASI input.
CASIO	Column Address Strobe Input, connected to CASI Input of Bank-0 Am29C668
CASI1	Column Address Strobe Input, connected to CASI Input of Bank-1 Am29C668
CA0	Column Address bit 0 for Bank 0
CA1	Column Address bit 0 for Bank 1

CASSEL0	CAS0 Select, internal signal used to generate interleaved CASI0 signal
CASSEL1	CAS1 Select, internal signal used to generate interleaved CASI1 signal
RAS_EN	RAS Enable, internal signals used to enable RASI

The RASI output is connected to the Am29C668 RASI input and is used to control the **RASn** outputs. RASI starts all memory accesses, Read/Writes, refresh and reset cycles. RASI can only be asserted during the transition from the idle state to the first wait state. Since RASI is asynchronous, **RAS_EN** is used to insure that RASI is asserted at this time. **RAS_EN** is asserted when **CLK** is Low during the idle state and **RFRQ**, **CS** or **CS_RESET** is asserted. **RAS_EN** is latched on the rising edge of **CLK**. Once RASI is asserted, it remains asserted until one of following conditions happens: a refresh cycle is requested or a cache miss occurs. Figure 2C shows that a cache miss can only occur during the **PD_ST**. This causes RASI to be deasserted, pre-charging the **RAS** lines in the DRAMs. A refresh request can occur during any cycle, but it does not affect the state machine until it is in the **PWAIT_ST** or **PD_ST**. If **RFRQ** is asserted during either of these states, RASI is deasserted to precharge **RAS** for the refresh cycle.

CASI0 and CASI1, connected to the CASI inputs of Bank 0 and Bank 1 respectively are used to access the banks in an interleaved manner. **BANK_PTR** determines the currently active bank; if **BANK_PTR** is High, Bank 1 is active, otherwise Bank 0 is active. During the **WS2_ST** or **PD_ST**, CASI to the active bank is asserted when **CLK** goes Low. CASI remains active for the next 1 - 1/2 cycles. This completes the access to that bank. During any Read access, the CASI to the inactive bank is asserted one clock cycle (40 ns in this case) after the CASI to the active bank. This CASI overlapping hides part of the access to the successive bank, thus diminishing the system access time to memory. This overlapping of accesses results in a single-cycle burst access. **CASSEL0** is used to turn on CASI0 to Bank 0 and to keep it on while the **BANK_PTR** is changing values. **CASSEL1** performs the same function for CASI1.

CA0 and CA1 are the lowest order bits to the DRAMs. These two outputs along with **BANK_PTR** implement the burst counter. Because the 80486 uses a different method to increment the address than the Am29C668 column counter, the internal column counter cannot be used for this design. Instead, an external counter must be implemented. As a result, ALE must be tied High so that the counter can update the address to the DRAMs. Table 1 shows all the possible combinations of burst memory accesses. From this table, it is evident that each subsequent access is to a different bank and that each subsequent access to the same bank differs by one bit. This means A2 determines which bank is active and A3 determines which word within that bank is ac-

cessed. The next address is generated by properly negating the bits at the appropriate time. A2 is negated on every access and A3 is negated every other access; this is the same as negating bit A3 to each bank separately after the bank has been accessed. **BANK_PTR** is then set to A2 at the beginning of the access and both RA0 and RA1 are set to A3. **BANK_PTR** is negated only during the **ACC_ST**. RA0, and RA1 are negated after the address has been latched by the DRAMs to provide ample time for the new address to propagate to the DRAMs and satisfy the set-up time of the column address relative to **CAS**.

CA0 and CA1 select between A3 and RA[1:0], depending upon the current state, so that the address can propagate to the DRAMs early, during accesses to the currently active page. If this were not done, an extra cycle would be needed during these cycles, negating some of the advantages of using the Am29C668 cache mode.

State PAL Device

The State PAL device is responsible for arbitrating between memory accesses and for implementing the memory state machine. It generates the Cache Enable signal, the Ready and the Burst Ready signals to the 80486. The following inputs are used.

CLK	25-MHz system Clock
CS	Chip Select, from the Address Decoder
CS_RESET	Chip Select Reset, from the Address Decoder
CS_PRG	Chip Select Program, from the Address Decoder
BLAST	Burst-Last output, from the 80486
CH	Cache Hit, from the Am29C668
RRFQ	Refresh Request, from the RAS and CAS PAL device
RFCYC	Refresh Cycle, from the Control PAL device
RESET	Reset, from the Reset Circuitry
BANK_PTR	Bank Pointer, from the Control PAL device
MC0	Mode Control 0, from the Control PAL device
W_R	Write active High, Read active Low, from the 80486

The following outputs are generated:

STATE[2:0]	State Variables, indicate current memory state
OE[1:0]	Data buffer Output Enables
KEN	Cache Enable signal, to 80486

RDY	Ready, to 80486
BRDY	Burst Ready, to 80486

The **KEN** output is used by the 80486 to determine which memory Reads can be stored in the on-board 8-Kbyte cache. It must be asserted at least one cycle before **RDY** or **BRDY** is returned. **KEN** must also be valid in the second to last cycle of the burst access, so that the 80486 can validate the cache line. In this application, **KEN** is asserted at the beginning of any Read access to the memory space and remains asserted until the access terminates. If there are multiple memory spaces, **KEN** can be driven from several sources; the multiple **KEN** signals must be combined in one PAL to generate a single **KEN** signal to the 80486. For high-speed designs, this approach is better than using multiple wired-OR outputs. This is also applicable to **RDY** and **BRDY**.

The **RDY** and **BRDY** signals to the 80486 indicate that data is valid from the memory during Read cycles, or that the data has been written to memory during Write cycles. **RDY** is returned during non-burst cycles. **BRDY** is returned to indicate that the access is a burst cycle and that the next access can complete in one cycle. If **BRDY** is not returned, the next data cannot be read until two cycles after the cycle in which **RDY** was asserted. If the processor expects **RDY**, for simple accesses or the last cycle of a burst access, and the memory returns **BRDY**, the processor treats it the same as if **RDY** were asserted. For systems with multiple external devices, the **RDY** from each device should be ORed together to produce one **RDY** signal to the 80486. The same applies for **BRDY**.

The **OE[1:0]** outputs control the output drivers of the data buffers that drive the Read data on to the CPU data bus.

The **STATE[2:0]** outputs are the state variables for the memory state machine; seven states are used (Figure 3). After Reset, the state machine is always in the idle state **IDLE_ST**, where it remains until there is a valid access to the memory. There are four major types of memory accesses: Read/Write, refresh, program and software reset. The state machine provides arbitration among these four accesses. If the memory is in the **IDLE_ST**, refresh requests are given highest priority to insure memory integrity. During a memory access, a burst is completed before a refresh request is serviced to minimize the memory access overhead to the 80486 in burst mode. In addition, since the 80486 has a fixed burst length of four words, the refresh cycle is delayed by a fixed maximum time that must be taken into account in the design of the refresh counter.

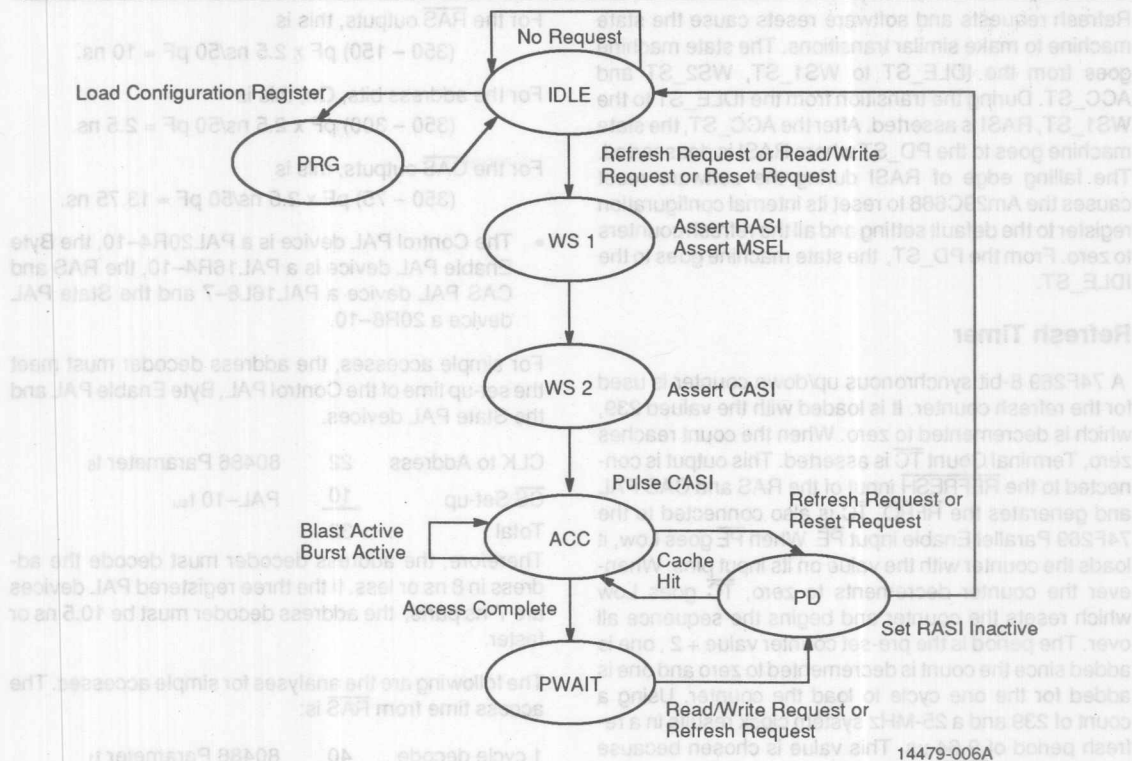


Figure 3. State Machine

An initial Read/Write access begins in the IDLE_ST. From the IDLE_ST the state machine goes to WS1_ST, WS2_ST and then ACC_ST. This completes the first access. During the transition from the IDLE_ST to the WS1_ST, RASI is asserted. In the second half of the WS1_ST, MSEL is asserted. In the second half of the WS2_ST, the appropriate CASI is asserted. If the access is a Write access, the state machine goes from the ACC_ST to the PWAIT_ST, if there are no refresh requests pending. If the access is a simple Read request, the state machine goes to the PWAIT_ST if there are no refresh requests pending. If a refresh request is pending, the state machine goes to the PD_ST; RASI is deasserted in the PD_ST and remains so in the IDLE_ST. The state machine then behaves as described in the paragraph below. If the Read request is a burst request, the state machine remains in the ACC_ST until the 80486 asserts $\overline{\text{BLAST}}$. The state machine does not keep track of the number of burst accesses, but requires the 80486 to do so. Once the burst is complete, the state machine goes to the PWAIT_ST unless there is an outstanding refresh request. In the PWAIT_ST, the state machine waits for a memory request. Once one is received, the state machine always goes to the PD_ST. In the PD_ST, the state machine arbitrates between refresh requests and processor re-

quests. In this case, refresh requests receive priority. During refresh request, the state machine goes to the IDLE_ST state and follow the description below. If the processor requests a program cycle or a software reset, the state machine also goes to the IDLE_ST state and behaves as described below. If a memory access is requested and there are no outstanding refresh requests, the state machine goes to the ACC_ST if $\overline{\text{CH}}$ is asserted. If $\overline{\text{CH}}$ is not asserted, a page miss has occurred, and the state machine must go to the IDLE_ST and perform an initial access. RASI is deasserted on a page miss as soon as the state variables change to the PD_ST to provide ample time for pre-charging RAS. If a page hit occurs, the appropriate CASI is asserted during the second half of the PD_ST and the access is completed as described above.

A program cycle causes the state machine to go from the IDLE_ST to the program state PRG_ST. $\overline{\text{RDY}}$ is returned to the 80486 in the PRG_ST and MSEL_RL is asserted during the second half of the PRG_ST, loading the configuration register. The state machine then returns to the IDLE_ST. Program cycles require at least one cycle to complete.

Refresh requests and software resets cause the state machine to make similar transitions. The state machine goes from the IDLE_ST to WS1_ST, WS2_ST and ACC_ST. During the transition from the IDLE_ST to the WS1_ST, RASI is asserted. After the ACC_ST, the state machine goes to the PD_ST where RASI is deasserted. The falling edge of RASI during the software reset causes the Am29C668 to reset its internal configuration register to the default setting and all the refresh counters to zero. From the PD_ST, the state machine goes to the IDLE_ST.

Refresh Timer

A 74F269 8-bit synchronous up/down counter is used for the refresh counter. It is loaded with the value 239, which is decremented to zero. When the count reaches zero, Terminal Count \overline{TC} is asserted. This output is connected to the REFRESH input of the RAS and CAS PAL and generates the \overline{RFRQ} . \overline{TC} is also connected to the 74F269 Parallel Enable input \overline{PE} . When \overline{PE} goes Low, it loads the counter with the value on its input pins. Whenever the counter decrements to zero, \overline{TC} goes Low which resets the counter and begins the sequence all over. The period is the pre-set counter value + 2, one is added since the count is decremented to zero and one is added for the one cycle to load the counter. Using a count of 239 and a 25-MHz system clock results in a refresh period of 9.64 μ s. This value is chosen because the maximum \overline{RAS} active time is 10 μ s for normal access. The refresh counter insures that this parameter is never violated.

Data Buffers

Am29C863A Bus Transceivers are used for the data buffers: four per bank are needed or eight for this memory system. The Am29C863A offers a simple interface requiring only two output enable signals and supports parity with its 9-bit port.

Timing Analyses

This design must meet many timing constraints. There are two different modes of operation, simple and burst accesses. Timing for the simple accesses is discussed first. Note: All timings in this section are in nanoseconds unless otherwise stated.

The following assumptions are used in these memory calculations:

- 9-bit SIMM modules are used.
- Input capacitance of the address, \overline{RAS} and \overline{CAS} is 75 pF per module.
- Since the load capacitances are less than those specified in the data sheet, they must be reduced by the derating factor of 2.5 ns/50 pF.

For the \overline{RAS} outputs, this is

$$(350 - 150) \text{ pF} \times 2.5 \text{ ns/50 pF} = 10 \text{ ns.}$$

For the address bits, Q_n , this is

$$(350 - 300) \text{ pF} \times 2.5 \text{ ns/50 pF} = 2.5 \text{ ns.}$$

For the \overline{CAS} outputs, this is

$$(350 - 75) \text{ pF} \times 2.5 \text{ ns/50 pF} = 13.75 \text{ ns.}$$

- The Control PAL device is a PAL20R4-10, the Byte Enable PAL device is a PAL16R4-10, the RAS and CAS PAL device a PAL16L8-7 and the State PAL device a 20R8-10.

For simple accesses, the address decoder must meet the set-up time of the Control PAL, Byte Enable PAL and the State PAL devices.

CLK to Address	22	80486 Parameter t_6
\overline{CS} Set-up	10	PAL-10 t_{su}
Total	32	

Therefore, the address decoder must decode the address in 8 ns or less. If the three registered PAL devices are 7-ns parts, the address decoder must be 10.5 ns or faster.

The following are the analyses for simple accesses. The access time from \overline{RAS} is:

1 cycle decode	40	80486 Parameter t_1
CLK to RASI	7.5	PAL16L8-7 t_{pd}
RASI to \overline{RAS}_n	16	Am29C668-1 Parameter 17
\overline{RAS} to Data	80	DRAM t_{rac}
Buffer Delay	6.5	Am29C861A t_{pd}
Data Setup	5	80486 Parameter t_{22}
Total	155	

The access time from \overline{CAS} is:

2 cycles	80	80486 Parameter t_1
CLK High max	24	80486 Parameter t_2
CLK to CASI	7.5	PAL16L8-7 t_{pd}
CASI to \overline{CAS}_n	13	Am29C668-1 Parameter 18
\overline{CAS}_n to Data	20	DRAM t_{cac}
Buffer Delay	6.5	Am29C861A t_{pd}
Data Setup	5	80486 Parameter t_{22}
Total	15	

The access time from the column address is:

1 cycle	40	80486 Parameter t_1
CLK High max	24	80486 Parameter t_2
CLK to MSEL	10	PAL20R6-10 t_{pd}
MSEL to Q_n	23.5	Am29C668 Parameter 19
Q_n to Data	40	DRAM t_{aa}
Buffer Delay	6.5	Am29C861A t_{pd}

Data Setup	<u>5</u>	80486 Parameter t ₂₂
Total	<u>149</u>	

From these three calculations, the initial access is completed in four cycles.

For cache hit accesses, the following accesses apply. Access time from CAS is:

1 cycle decode	40	80486 Parameter t ₁
CLK High max	24	80486 Parameter t ₂
CLK to CASI	7.5	PAL16L8-7 t _{pd}
CASI to CASn	13	Am29C668 Parameter 18
CASn to Data	20	DRAM t _{cac}
Buffer Delay	6.5	Am29C861A t _{pd}
Data Setup	<u>5</u>	80486 Parameter t ₂₂
Total	<u>116</u>	

The access time from the column address is:

CLK to address	22	80486 Parameter t ₆
Address to Qn	26.5	Am29C668 Parameter 1
Qn to Data	40	DRAM t _{aa}
Buffer Delay	6.5	Am29C861A t _{pd}
Data Setup	<u>5</u>	80486 Parameter t ₂₂
Total	<u>100</u>	

Therefore, the cache mode access requires only three cycles to complete.

The following timings apply to the burst access. The access time from CAS is the same as the calculation in the cache mode access, except that there is no decoding. The address is incremented for the next access one cycle before CASI is asserted to provide an extra cycle for the address to propagate to the DRAMs.

CLK to RA	8	PAL20R8-10 t _{coo}
RA to CA	7.5	PAL16L8-7 delay
ACn to Qn	<u>26.5</u>	Am29C668 Parameter 1
Total	<u>42</u>	

The column address is valid 2 ns into the cycle in which CASI is asserted; therefore the access time from the column address is more than satisfied and a burst access is completed in two cycles. With interleaving, the effective burst rate is one cycle per clock.

PARTS LIST

Part	Count
PALCE16V8	1
PALCE20V8	3
Am29C668	2
Am29C863A	8
74F269	1
Decoder	1
Memories	64
25 MHz Osc.	<u>1</u>
Total	<u>81</u>

PAL Equations

The following application notes are guides to interfacing the Am29C668 with popular microprocessors. They are paper designs only, and have not been built and tested.

The following are the logic equations for the PALs. They are written in PALASM.

;PALASM Design Description

; Declaration Segment

TITLE RAS AND CAS

PATTERN A

REVISION 1.0

COMPANY AMD INC.

DATE 07/12/91

CHIP _RCAS486 PALCE20V8

; PIN Declarations

PIN 1	CLK	COMBINATORIAL ; INPUT
PIN 2	/CS	COMBINATORIAL ; INPUT
PIN 3	/CS_RESET	COMBINATORIAL ; INPUT
PIN 4	/RFRQ	COMBINATORIAL ; INPUT
PIN 5	/RFCYC	COMBINATORIAL ; INPUT
PIN 6	/MSTATE[2]	COMBINATORIAL ; INPUT
PIN 7	/MSTATE[1]	COMBINATORIAL ; INPUT
PIN 8	/MSTATE[0]	COMBINATORIAL ; INPUT
PIN 9	/CH	COMBINATORIAL ; INPUT
PIN 10	W_RL	COMBINATORIAL ; INPUT
PIN 11	RA[1]	COMBINATORIAL ; INPUT
PIN 13	RA[0]	COMBINATORIAL ; INPUT
PIN 14	A[3]	COMBINATORIAL ; INPUT
PIN 15	CA1	COMBINATORIAL ; OUTPUT
PIN 16	/RAS_EN	COMBINATORIAL ; OUTPUT
PIN 17	/CASSEL1	COMBINATORIAL ; OUTPUT
PIN 18	/CASSEL0	COMBINATORIAL ; OUTPUT
PIN 19	CAS11	COMBINATORIAL ; OUTPUT
PIN 20	CAS10	COMBINATORIAL ; OUTPUT
PIN 21	RAS1	COMBINATORIAL ; OUTPUT
PIN 22	CA0	COMBINATORIAL ; OUTPUT
PIN 23	/BANK_PTR	COMBINATORIAL ; OUTPUT
PIN 12	GND	; INPUT
PIN 24	VCC	; INPUT

;SPECIAL DEFINITIONS

STRING IDLE_ST '/MSTATE[2] * /MSTATE[1] * /MSTATE[0]'

STRING WS1_ST '/MSTATE[2] * /MSTATE[1] * MSTATE[0]'

STRING WS2_ST '/MSTATE[2] * MSTATE[1] * MSTATE[0]'

STRING ACC_ST '/MSTATE[2] * MSTATE[1] * /MSTATE[0]'

STRING PWAIT_ST 'MSTATE[2] * MSTATE[1] * MSTATE[0]'

```

STRING PD_ST 'MSTATE[2] * MSTATE[1] * /MSTATE[0]'
STRING PRG_ST 'MSTATE[2] * /MSTATE[1] * /MSTATE[0]'
;----- Boolean Equation Segment -----
EQUATIONS
CA0 = A[3] * (PWAIT_ST + WS1_ST + WS2_ST + PD_ST) + ACC_ST * RA[0]
CA1 = A[3] * (PWAIT_ST + WS1_ST + WS2_ST + PD_ST) + ACC_ST * RA[1]
IF (/MSTATE[2] * /MSTATE[1] * RAS_EN * CLK) THEN
  BEGIN
    RASI = 1
  END
ELSE
  BEGIN
    IF (PWAIT_ST * RFRQ + PD_ST * (CS * /CH + RFRQ) + PRG_ST) THEN
      BEGIN
        RASI = 0
      END
    ELSE
      BEGIN
        RASI = RASI
      END
    END
  END
CASIO = CASSEL0 * /CLK + ACC_ST * /CLK * /W_RL + CASIO * (CASSEL0 + /CLK * /MSTATE[2] * MSTATE[1])
CASII = CASSEL1 * /CLK + ACC_ST * /CLK * /W_RL + CASII * (CASSEL1 + /CLK * /MSTATE[2] * MSTATE[1])
CASSEL0 = (WS2_ST + PD_ST * CH * /RFRQ) * /BANK_PTR + WS2_ST * RFCYC + ACC_ST * BANK_PTR * /CLK
          + CASSEL0 * CLK
CASSEL1 = (WS2_ST + PD_ST * CH * /RFRQ) * BANK_PTR + WS2_ST * RFCYC + ACC_ST * /BANK_PTR *
          /CLK + CASSEL1 * CLK
RAS_EN = PD_ST * CH * /RFRQ + IDLE_ST * (RFRQ + CS + CS_RESET) * /CLK + RAS_EN * CLK
;----- Simulation Segment -----
SIMULATION
;

```

;PALASM Design Description

;----- Declaration Segment -----

TITLE CONTROL

PATTERN A

REVISION 1.0

COMPANY AMD INC.

DATE 07/12/91

CHIP _CONT486 PALCE20V8

;----- PIN Declarations -----

PIN 1	CLK	COMBINATORIAL ; INPUT
PIN 2	CLK_COMB	COMBINATORIAL ; INPUT
PIN 3	/CS	COMBINATORIAL ; INPUT
PIN 4	/CS_RESET	COMBINATORIAL ; INPUT
PIN 5	/CS_PRG	COMBINATORIAL ; INPUT
PIN 6	W_RL	COMBINATORIAL ; INPUT
PIN 7	/MSTATE[2]	COMBINATORIAL ; INPUT
PIN 8	/MSTATE[1]	COMBINATORIAL ; INPUT
PIN 9	/MSTATE[0]	COMBINATORIAL ; INPUT
PIN 10	/REFRESH	COMBINATORIAL ; INPUT
PIN 11	A[2]	COMBINATORIAL ; INPUT
PIN 13	/OE	COMBINATORIAL ; INPUT
PIN 14	A[3]	COMBINATORIAL ; INPUT
PIN 23	RASI	COMBINATORIAL ; INPUT
PIN 15	/RFRQ	COMBINATORIAL ; OUTPUT
PIN 16	MC0	COMBINATORIAL ; OUTPUT
PIN 17	/BANK_PTR	REGISTERED ; OUTPUT
PIN 18	/OE_DIN	REGISTERED ; OUTPUT
PIN 19	RA[0]	REGISTERED ; OUTPUT
PIN 20	RA[1]	REGISTERED ; OUTPUT
PIN 21	/RFCYC	COMBINATORIAL ; INPUT
PIN 22	MSEL_RL	COMBINATORIAL ; OUTPUT
PIN 12	GND	; INPUT
PIN 24	VCC	; INPUT

;SPECIAL DEFINITIONS

STRING IDLE '#B000'

STRING WS1 '#B001'

STRING WS2 '#B011'

STRING ACC '#B010'

STRING PWAIT '#B111'

STRING PD '#B110'

STRING IDLE_ST '/MSTATE[2] * /MSTATE[1] * /MSTATE[0]'

STRING WS1_ST '/MSTATE[2] * /MSTATE[1] * MSTATE[0]'

STRING WS2_ST '/MSTATE[2] * MSTATE[1] * MSTATE[0]'

STRING ACC_ST '/MSTATE[2] * MSTATE[1] * /MSTATE[0]'

```
STRING PD_ST 'MSTATE[2] * MSTATE[1] * /MSTATE[0]'
```

```
STRING PRG_ST 'MSTATE[2] * /MSTATE[1] * /MSTATE[0]'
```

```
;----- Boolean Equation Segment -----
```

```
EQUATIONS
```

```
MSEL_RL = PRG_ST * CLK_COMB * MC0 + MSEL_RL * PRG_ST + RASI * /CLK_COMB + MSEL_RL * RASI
```

```
MC0 = IDLE_ST * (CS_PRG + CS_RESET) + MC0 * (WS1_ST + WS2_ST + ACC_ST + PD_ST + PRG_ST)
```

```
OE_DIN := W_RL * CS
```

```
CASE (MSTATE[2..0])
```

```
  BEGIN
```

```
    IDLE:
```

```
      BEGIN
```

```
        IF (CS) THEN
```

```
          BEGIN
```

```
            BANK_PTR := A[2]
```

```
          END
```

```
        END
```

```
      WS1:
```

```
        BEGIN
```

```
          BANK_PTR := BANK_PTR
```

```
        END
```

```
      WS2:
```

```
        BEGIN
```

```
          BANK_PTR := BANK_PTR
```

```
          IF (/RFCYC * /BANK_PTR) THEN
```

```
            BEGIN
```

```
              RA[0] := /A[3]
```

```
              RA[1] := A[3]
```

```
            END
```

```
          ELSE
```

```
            BEGIN
```

```
              RA[0] := A[3]
```

```
              RA[1] := /A[3]
```

```
            END
```

```
          END
```

```
      ACC:
```

```
        BEGIN
```

```
          IF (/RFCYC * /MC0) THEN
```

```
            BEGIN
```

```
              BANK_PTR := /BANK_PTR
```

```
              IF (/BANK_PTR) THEN
```

```
                BEGIN
```

```
                  RA[0] := RA[0]
```

```
                  RA[1] := /RA[1]
```

```
                END
```



```

ELSE
    BEGIN
        RA[1] := RA[1]
        RA[0] := /RA[0]
    END
END
PWAIT:
BEGIN
    IF (CS * /RFRQ) THEN
        BEGIN
            MSTATE[2..0] := PD
            BANK_PTR := A[2]
        END
    END
PD:
BEGIN
    BANK_PTR := BANK_PTR
    IF (/BANK_PTR) THEN
        BEGIN
            RA[0] := /A[3]
            RA[1] := A[3]
        END
    ELSE
        BEGIN
            RA[1] := /A[3]
            RA[0] := A[3]
        END
    END
END
END ;"CASE"
;
SIMULATION
;

```

Declaration Segment

- PIN Declarations

;PALASM Design Description

;-----Declaration Segment-----

TITLE MEMORY STATE MACHINE

PATTERN A

REVISION 1.0

COMPANY AMD INC.

DATE 07/12/91

CHIP _STATE486 PALCE20V8

;-----PIN Declarations-----

PIN 1	CLK	COMBINATORIAL ; INPUT
PIN 2	/CS	COMBINATORIAL ; INPUT
PIN 3	/CS_PRG	COMBINATORIAL ; INPUT
PIN 4	/CS_RESET	COMBINATORIAL ; INPUT
PIN 5	/BLAST	COMBINATORIAL ; INPUT
PIN 6	/CH	COMBINATORIAL ; INPUT
PIN 7	W_RL	COMBINATORIAL ; INPUT
PIN 8	/RFRQ	COMBINATORIAL ; INPUT
PIN 9	/RFCYC	COMBINATORIAL ; INPUT
PIN 10	RESET	COMBINATORIAL ; INPUT
PIN 11	/BANK_PTR	COMBINATORIAL ; INPUT
PIN 13	/OE	COMBINATORIAL ; INPUT
PIN 14	MC0	COMBINATORIAL ; INPUT
PIN 15	/KEN	REGISTERED ; OUTPUT
PIN 16	/RDY	REGISTERED ; OUTPUT
PIN 17	/BRDY	REGISTERED ; OUTPUT
PIN 18	/OE[0]	REGISTERED ; OUTPUT
PIN 19	/OE[1]	REGISTERED ; OUTPUT
PIN 20	/MSTATE[0]	REGISTERED ; OUTPUT
PIN 21	/MSTATE[1]	REGISTERED ; OUTPUT
PIN 22	/MSTATE[2]	REGISTERED ; OUTPUT
PIN 12	GND	; INPUT
PIN 24	VCC	; INPUT

;SPECIAL DEFINITIONS

STRING IDLE '#B000'

STRING WS1 '#B001'

STRING WS2 '#B011'

STRING ACC '#B010'

STRING PWAIT '#B111'

STRING PD '#B110'

STRING PRG '#B100'

STRING UN1 '#B101'

STRING IDLE_ST '/MSTATE[2] * /MSTATE[1] * /MSTATE[0]'

STRING WS1_ST '/MSTATE[2] * /MSTATE[1] * MSTATE[0]'

STRING WS2_ST '/MSTATE[2] * MSTATE[1] * MSTATE[0]'

```

STRING ACC_ST '/MSTATE[2] * MSTATE[1] * /MSTATE[0]'
STRING PWAIT_ST 'MSTATE[2] * MSTATE[1] * MSTATE[0]'
STRING PD_ST 'MSTATE[2] * MSTATE[1] * /MSTATE[0]'
STRING PRG_ST 'MSTATE[2] * /MSTATE[1] * /MSTATE[0]'
STRING UN1_ST 'MSTATE[2] * /MSTATE[1] * MSTATE[0]'
;----- Boolean Equation Segment -----

```

EQUATIONS

IF (RESET) THEN

BEGIN

MSTATE[2..0] := IDLE

BRDY := 0

RDY := 0

OE[1] := 0

OE[0] := 0

END

CASE (MSTATE[2..0])

BEGIN

IDLE:

BEGIN

IF (RFRQ) THEN

BEGIN

MSTATE[2..0] := WS1

END

ELSE

BEGIN

IF (CS_PRG) THEN

BEGIN

MSTATE[2..0] := PRG

RDY := 1

END

ELSE

BEGIN

IF (CS + CS_RESET) THEN

BEGIN

MSTATE[2..0] := WS1

IF (CS * /W_RL) THEN

BEGIN

KEN := 1

END

END

ELSE

BEGIN

MSTATE[2..0] := IDLE

END

END	STRING ACC_ST 'MSTATE[1] * MSTATE[2] * MSTATE[0]
END	STRING PWAIT_ST 'MSTATE[1] * MSTATE[2] * MSTATE[0]
END	STRING PD_ST 'MSTATE[1] * MSTATE[2] * MSTATE[0]
WS1:	STRING PRO_ST 'MSTATE[1] * MSTATE[2] * MSTATE[0]
BEGIN	STRING UNI_ST 'MSTATE[1] * MSTATE[2] * MSTATE[0]
MSTATE[2..0] := WS2	Boolean Equation System
IF (/W_RL * /RFCYC) THEN	EQUATIONS
BEGIN	IF (RESET) THEN
KEN := 1	BEGIN
END	MSTATE[2..0] := IDLE
END	BRDY := 0
WS2:	RDY := 0
BEGIN	OE[1] := 0
MSTATE[2..0] := ACC	OE[0] := 0
IF (/W_RL * /RFCYC) THEN	END
BEGIN	CASE (MSTATE[0])
KEN := 1	BEGIN
END	IDLE:
IF (/RFCYC) THEN	BEGIN
BEGIN	IF (PRQ) THEN
IF (/BLAST) THEN	BEGIN
BEGIN	MSTATE[2..0] := WS1
BRDY := 1	END
END	ELSE
ELSE	BEGIN
BEGIN	IF (CS_PRO) THEN
RDY := 1	BEGIN
END	MSTATE[2..0] := PRO
IF (/W_RL) THEN	RDY := 1
BEGIN	END
IF (BANK_PTR) THEN	ELSE
BEGIN	BEGIN
OE[1] := 1	IF (CS + CS_RESET) THEN
END	BEGIN
ELSE	MSTATE[2..0] := WS1
BEGIN	IF (CS * /W_RL) THEN
OE[0] := 1	BEGIN
END	KEN := 1
END	END
END	END
END	ELSE
ACC:	BEGIN
BEGIN	MSTATE[2..0] := IDLE
IF (/RFCYC * /BLAST * /MC0) THEN	END


```

BEGIN
  MSTATE[2..0] := ACC
  BRDY := 1
  KEN := 1
  IF (BANK_PTR) THEN
    BEGIN
      OE[0] := 1
    END
  ELSE
    BEGIN
      OE[1] := 1
    END
  END
END
ELSE
  BEGIN
    IF (RFRQ + RFCYC + MC0) THEN
      BEGIN
        MSTATE[2..0] := PD
      END
    ELSE
      BEGIN
        MSTATE[2..0] := PWAIT
        RDY := 0
      END
    END
  END
END
PWAIT:
  BEGIN
    IF (RFRQ) THEN
      BEGIN
        MSTATE[2..0] := PD
      END
    ELSE
      BEGIN
        IF (CS) THEN
          BEGIN
            MSTATE[2..0] := PD
            IF (W_RL) THEN
              BEGIN
                KEN := 1
              END
            END
          END
        ELSE
          BEGIN

```

```

MSTATE[2..0] := PWAIT
END
END
END
PD:
BEGIN
  IF (RFRQ + CS * /CH) THEN
    BEGIN
      MSTATE[2..0] := IDLE
    END
  ELSE
    BEGIN
      MSTATE[2..0] := ACC
      IF (W_RL) THEN
        BEGIN
          KEN := 1
          IF (BLAST) THEN
            BEGIN
              RDY := 1
            END
          ELSE
            BEGIN
              BRDY := 1
            END
          END
        END
      END
    END
  END
END
END
PRG:
BEGIN
  MSTATE[2..0] := IDLE
END
UN1:
BEGIN
  MSTATE[2..0] := IDLE
END
END ;"CASE"

```

```

;----- Simulation Segment -----

```

```

SIMULATION
;-----

```

CHAPTER 4

EDC Memory-Board Designs



Introduction	4-2
IBM PC-AT Plug-in Memory Card with EDC	4-3
IBM PS/2 12-Mbyte Memory Board with EDC	4-25

As systems require larger and larger memories, it is imperative to protect the memory from soft errors that occur when a single bit is complemented due to noise, alpha particles, or some other event. While single-bit errors are the most common, double- and multiple-bit errors sometimes occur. The Am95C88 EDC detects and corrects all single-bit errors and detects all double- and some multiple-bit errors. The Am95C88 CDMC can control large memories, up to four banks of 4-Mbit DRAMs, and can drive the RAS, CAS, and address lines without external drivers or damping resistors.



EDC Memory-Board Designs

INTRODUCTION

The following application notes describe evaluation boards that demonstrate the capabilities of the Am29C660C 32-bit Error Detection and Correction circuit (EDC) and the Am29C668 4M Configurable Dynamic Memory Controller (CDMC). The first board is IBM PC-AT compatible, the second is Micro-Channel and PS/2 compatible.

As systems require larger and larger memories, it is imperative to protect the memory from soft errors that occur when a single bit is complemented due to noise, alpha particles, or some other event. While single-bit errors are the most common, double- and multiple-bit errors sometimes occur. The Am29C660 EDC detects and corrects all single-bit errors and detects all double- and some multiple-bit errors. The Am29C668 CDMC can control large memories, up to four banks of 4-Mbit DRAMs, and can drive the \overline{RAS}_n , \overline{CAS}_n and address lines without external drivers or damping resistors.

IBM PC-AT Plug-in Memory Card with Error Detection and Correction (EDC)

INTRODUCTION

This PC-AT compatible evaluation board demonstrates the capabilities the Am29C668 Configurable Dynamic Memory Controller (CDMC) and the Am29C660C 32-bit Error Detection and Correction Circuit (EDC). As memory size and density increase, it becomes more important to protect the memory from soft errors. The EDC detects and corrects all single-bit errors and detects all double and some triple-bit errors. When a word is accessed, it is checked for errors and if an error is found, the corrected data is written back to memory as well as to the data bus. This board also performs memory "scrubbing", which is the detection and correction of single-bit errors during normal refresh cycles hidden from the microprocessor to maintain the integrity of seldom-accessed memory locations. Scrubbing the memory prevents accumulation of single-bit errors which, in turn, avoids most double-bit errors.

Note: an operating system that utilizes the upper 15 Mbytes in the PC-AT* address space, such as Xenix** or OS/2**, is required to effectively utilize the board. A simple memory test is provided.

Detailed schematics are included, beginning on page 17.

Distinctive Characteristics

- Corrects all single-bit errors, detects all double-bit and some triple-bit errors.
- 12 Mbyte of dynamic RAM(1 M x 1-bit packages).
- Am29C668 CDMC and Am29C660C EDC packaged in plastic leaded chip carriers for maximum component density.
- Supports memory scrubbing during refresh.
- Designed for 10 MHz systems.
- A Dynamic Memory Timing Controller implemented using Programmable Array Logic (PAL™) devices and delay lines.
- System Data Interface consists of two Am29C983s and one Am29C823.
- 32-bit internal data bus with 7-bit check bit and 7-bit syndrome bus.
- Syndrome latch for diagnostic and test purposes.
- Occupies the second through the 13th megabyte PC-AT address space memory block.

*PC-AT is a registered trademark of IBM Corporation.

**Xenix and OS/2 are trademarks of Microsoft Corporation.

DETAILED DESCRIPTION

The primary data paths and functional elements are shown in Figure 1. The following discussion describes each section of the block diagram in detail. Components not appearing in the block diagram but existing on the schematic are also discussed.

Edge Card Connectors

This board can only be used in a PC-AT backplane with the dual-connector I/O channel. Interrupt Request IRQ3 is used by the board to signal the detection of a multiple-bit error to the system microprocessor. Jumper W2 can be removed to prevent IRQ3 from being driven onto the backplane. All signals used from the edge connector are listed on page 6.

Memory Decoder

The 20L10B, chip U6, provides memory decoding for the board. It decodes the upper four bits, Local Addresses LA[23:20], to generate the four internal address bits, DMCA[23:20]. Table 1 shows the address translation. This address translation was chosen since most ATs contain 1 Mbyte of RAM on the mother board. A gap from the first to the second megabyte is required because the setup parameters of the host system (non-volatile RAM) are destroyed when the PC ROM routine BIOS detects RAM that was not properly installed by the "setup" program.

Table 1. Address Translation for the Memory Board

LA				DMCA				
23	22	21	20	23	22	21	20	
0	0	1	0	0	0	0	0	1st Row
0	0	1	1	0	0	0	1	
0	1	0	0	0	0	1	0	
0	1	0	1	0	0	1	1	
0	1	1	0	0	1	0	0	2nd Row
0	1	1	1	0	1	0	1	
1	0	0	0	0	1	1	0	
1	0	0	1	0	1	1	1	
1	0	1	0	1	0	0	0	3rd Row
1	0	1	1	1	0	0	1	
1	1	0	0	1	0	1	0	
1	1	0	1	1	0	1	1	

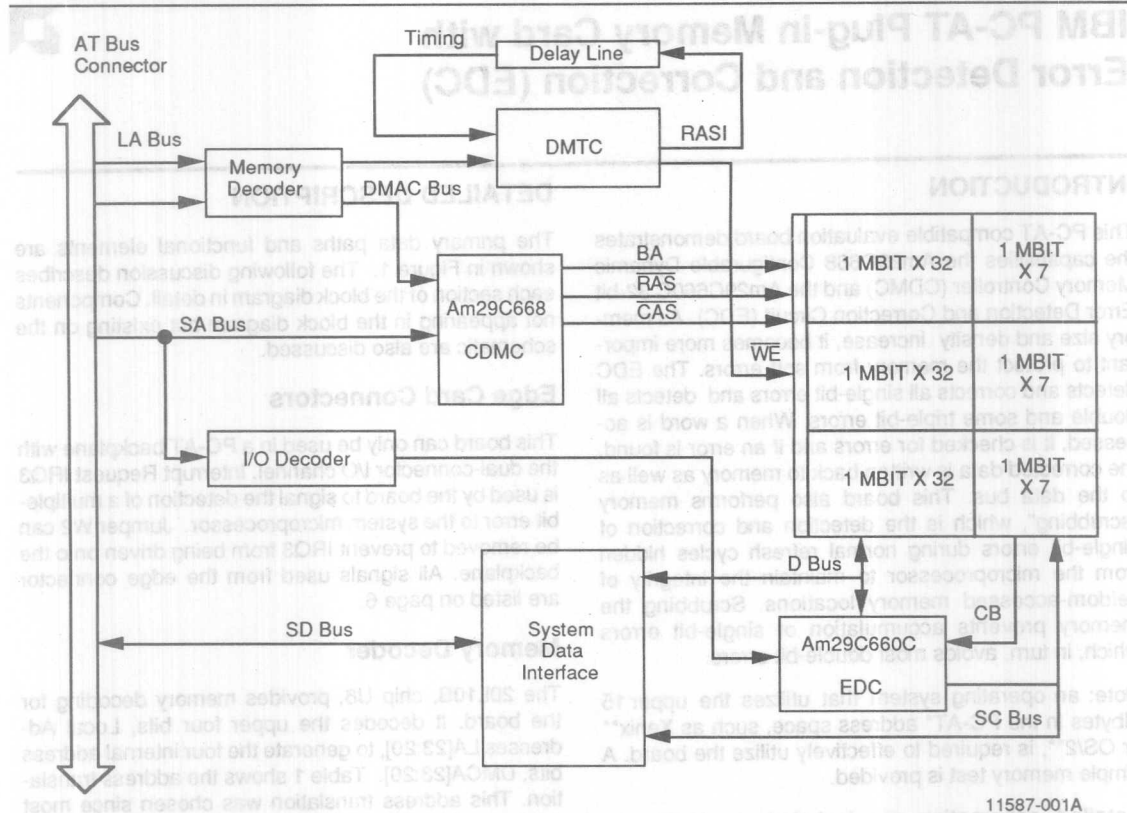


Figure 1. Block Diagram

The PAL U6 also generates the Cycle Request CYCREQ and Read/Write Request RWRQ signals when a valid memory access occurs. CYCREQ remains active until the processor completes its data transfer and RWRQ remains active until the full memory cycle is completed. CYCREQ and RWRQ are latched on Address Bus Latch Enable BALE going Low. The 16-Bit Memory Chip Select MEMCS16 is asserted during valid memory accesses and goes to a high-impedance state at all other times. The 16-bit Enable Memory Chip Select ENMCS16 controls the output signal of MEMCS16.

Refresh Request RRQ is generated during refresh cycles initiated by the processor driving REFRESH active. The Forced Refresh High FRH signal becomes active when REFRESH is High and is used to detect the falling edge of REFRESH. This is required because REFRESH is active longer than the memory refresh cycle.

I/O Channel Ready $\overline{\text{IOCHRDY}}$ is used to signal the processor that valid data is ready during Reads and to

signal completions of Writes. $\overline{\text{IOCHRDY}}$ is a 3-state output, enabled by $\overline{\text{CYCREQ}}$ when a valid memory access occurs. This board is designed to work in 10 MHz systems. If the board is used in faster systems (>12.5 MHz), a faster PAL may be necessary to assert $\overline{\text{IOCHRDY}}$ inactive so that the processor can detect the signal and insert wait states. This timing is machine dependent.

I/O Decoder

The I/O address decoding is provided by a combinatorial PAL, U7, and jumper W1. The Diagnostic Latch Enable LEDIAG and Syndrome Output Enable SOE are generated to select either the diagnostic latch or the syndrome latch respectively. The base address of the I/O ports is selectable by jumper W1 to be 320 h (Hex) or 220 h. Table 2 details this decoding scheme.

Table 2. I/O Decoding Scheme

System Address SA												$\overline{\text{IOW}}$	$\overline{\text{IOR}}$	Jumper W1	Port Decoded	Address
11	10	9	8	7	6	5	4	3	2	1	0					
0	0	1	0	0	0	1	0	0	0	0	0	1	0	in	$\overline{\text{SOE}}$	220 h
0	0	1	1	0	0	1	0	0	0	0	0	1	0	out	$\overline{\text{SOE}}$	320 h
0	0	1	0	0	0	1	0	0	0	1	0	0	1	in	LEDIAG	222 h
0	0	1	1	0	0	1	0	0	0	1	0	0	1	out	LEDIAG	322 h

Dynamic Memory Timing Control

The timing control for this board was implemented using PALs and delay lines for increased flexibility. The following subsections describe the signals and their functions.

RASI, Mode Control and End of Cycle

The Row Address Strobe Input RASI, Mode Control MC_n and End of Cycle $\overline{\text{EOC}}$ signals are generated by U8 in the Control Logic. RASI is used to initiate the timing sequence and to signal the CDMC, Unit 1, to generate the $\overline{\text{RASn}}$ signal to the appropriate bank of memory. The $\overline{\text{EOC}}$ signal is generated to signify the end of any memory cycle. This signal is used to reset the timing-tap outputs and place the internal logic into the initial state for the next memory access. The timing taps Tn are registered by a 20RA10, Unit 13. This technique provides shorter cycle times. Using a conventional design, it would be necessary to wait for the delay line to clear before the next cycle could start. The DONE signal is generated by the asynchronous PAL, U9, in the Control Logic to indicate that the required eight DRAM wake-up cycles have been completed and that the memory is ready for initialization. The MC_1 and MC_0 inputs of the CDMC determine which of the four operating modes will be used. Table 3 shows the decoding.

Latched Error, Initialization and Interrupt -3 Signals

Device U9 generates the $\overline{\text{LATCHED_ERR}}$ signal to indicate that an error has occurred on a Read or Read/Modify/Write cycle. This signal is set by the $\overline{\text{EOC}}$ signal and is sampled on the rising edge of timing tap T2. Every memory access is assumed to be a long cycle unless $\overline{\text{LATCHED_ERR}}$ is false. This assures correct and concise logic implementation. If $\overline{\text{LATCHED_ERR}}$ were conditionally set instead of reset, much more complicated logic would be required, since another signal is needed to indicate when $\overline{\text{LATCHED_ERR}}$ is valid.

Device U9 also latches the Initialize $\overline{\text{INIT}}$ signal, which is generated by the rising edge of RESET. $\overline{\text{INIT}}$ remains active until Terminal Count TC is received from the CDMC indicating that all the memory locations have been initialized. Counter[3:0] and DONE count the wake-up cycles and indicate when eight have been

Table 3. Mode Control Decoding

MC_1	MC_0	Mode
0	0	Refresh without scrubbing.
0	1	Refresh with scrubbing or initialize.
1	0	Read/Write mode.
1	1	Reset.

completed. Timing tap T8 indicates the end of a wake-up cycle and causes Counter[3:0] to increment. Counter[3:0] is initialized to 0 and, when it reaches 7, all eight cycles have been completed. DONE is asserted and the counting is inhibited. DONE remains active until $\overline{\text{INIT}}$ is deactivated.

Memory-Board Interrupt $\overline{\text{INTR3}}$ signals that a multiple error has occurred at T4. $\overline{\text{INTR3}}$ goes to Interrupt Request 3 IRQ3 on the backplane via jumper W2 and is cleared by an access to the syndrome latch (signal SOE). This signal can be disabled by removing Jumper W2. In systems that support bus retry registering, $\overline{\text{INTR3}}$ is not required.

Pulsed CASI

The Column Address Strobe Input CASI is a pulsed CAS line, used when connecting the data-in lines to the data-out lines on the DRAMs. The Interface Controller PAL, U11, generates this signal from the registered timing-tap signals received from U13, a 20RA10 PAL. Figure 2 shows how the pulsed CASI signal is produced. This only applies during Read/Modify/Write and Refresh with Scrubbing cycles. Note that at time A in the diagram, the DRAM outputs are 3-stated so that the Am29C660C can drive the data bus.

Miscellaneous Logic Functions

Miscellaneous logic functions are performed by the PAL20L8B Interface Controller, U12. $\overline{\text{WE}}[2:0]$ writes the data into the DRAMs. One signal per bank is necessary to drive the large capacitive load (273 pF). These signals have 39 Ω series damping resistors to control over- and under-shoot. Latch Enable Out/Generate LEO_GEN is a dual-purpose signal. When active High, it enables the output latches of the Am29C660C.

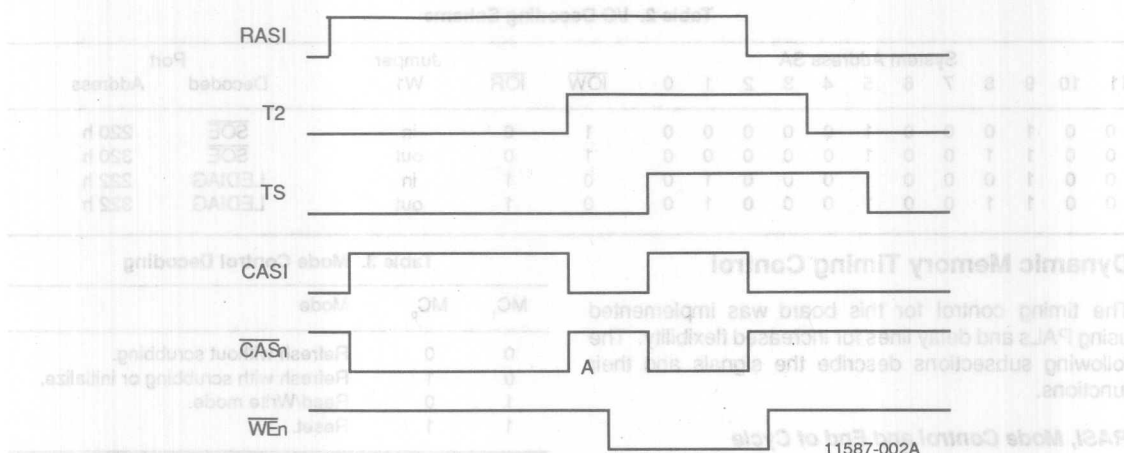


Figure 2. Idealized Timing Diagram for Pulsed CASI Signal

When active Low, it causes check bits to be generated for the data in the input latch of the Am29C660C. The Latch Enable Input LEI controls the latching of data into the 29C660C. When LEI is High, the input latch is transparent; data is latched when LEI is Low. Byte Latch Enable LEB controls the data latch from the internal data bus to the system bus. When LEB is High, the latch is transparent. S_AND_LEB is used to condition the output enables from the system bus.

Delay Lines

The delay lines, U14 through U17, provide the timing reference signals. RAS_i initiates the timing sequence. See Delay-Line Tap Calculations, page 15, for the tap-timing calculations of the timing configuration currently installed on the board. According to the board timing requirements, each timing signal must be reset at the beginning of each cycle. Device U13 is used to register the timing taps. Since it has separate clocks for each of its 10 output registers, this PAL saves board space over discrete logic. The outputs are reset on EOC going active. Using the technique of registering the timing-tap output shortens the memory cycle time.

Interface Control

PAL U10 generates the Output Enable Byte Low \overline{OE}_{BL} and the Output Enable Byte High \overline{OE}_{BH} signals, used by the respective 29C983 Multiple Bus Exchange device, Units 3 and 4, to output the proper data word (16 bits). When address bit SA_i is Low, the lower word (16-bits) is selected; when High, the upper word is selected. In addition, PAL U10 generates the Output Enable signals, $\overline{OE}[3:0]$, to the EDC and to PAL U11 that control byte selection in byte and word writes. SA₀₋₁

and \overline{SBHE} are decoded so that the selected byte or word is written to memory.

PAL U11 generates Syndrome Latch Enable \overline{SYN}_{EN} and Output Byte Enable $\overline{OE}_{BYTE}[3:0]$. Syndrome Latch Enable \overline{SYN}_{EN} is an active Low signal that latches the syndrome bits when an error is detected. Device U11 drives the $\overline{OE}_{BYTE}[3:0]$ inputs of the 29C983 latched transceivers that provide the system data-bus interface. These signals gate the AT-bus data lines to and from the various byte-wide data bits of the internal data bus on the board (the D bus). Note from the PAL equations that the gating signals are conditioned by S_AND_LEB to ensure proper latching of the data from the DRAMs. LG enables the latches.

Configurable Dynamic Memory Controller and Buffers

The Am29C668 CDMC, U1, supplies the DRAM array with multiplexed row and column address signals, as well as RAS_n and CAS_n lines. Timing inputs to this device are provided by the delay lines. The Am29C668 is used in the Am29368-compatible mode for 1 Mbit DRAMs. During Initialization, the Am29C668 generates initialization cycles until the entire memory is written with data and check bits. When the initialization is complete, TC is asserted High. \overline{RAS} -only Refresh is used when no memory scrubbing is selected; jumper W3 is connected. Note: AC₁₀ is not connected since 1 Mbit DRAMs are used. AC₁₀ is the most significant bit of the column latch and would be output on Q₁₀. Since 1 Mbit DRAMs only use Q₀₋₉, this address bit would be lost and the data would be written into the wrong location. 1 Mbit DRAMs use AC₀₋₉ for the column address and AR₀₋₉ for the row address.

Error Detection and Correction Circuit

Device U2, a high-speed Am29C660C EDC, generates check bits during a write and verifies the data and check bits during a read. Separate error ($\overline{\text{ERR}}$) and multiple-bit error ($\overline{\text{MERR}}$) signals are output if one or more errors are detected. If $\overline{\text{MERR}}$ is asserted, IRQ_3 is active if enabled by jumper W2. Jumper W4 selects input signal CODEID₀ to U2. If W4 is installed, a 32-bit slice is selected (the chip may operate in 32- or 64-bit mode); if W2 is not installed, the chip operates in internal-control mode. The internal-control mode provides access to the diagnostic registers in the Am29C660 and aids in debugging and testing the board. See the Am29C660C data sheet for further details.

The Am29C660C internal diagnostic latch is available from the I/O channel. To write the diagnostic register, a word is written from the AT microprocessor to the address selected by the I/O decoder and associated jumper. See I/O Decoder section.

The Output Enable Syndrome/Check Bit $\overline{\text{OESC}}$ pin on the EDC is grounded to eliminate the T7-to-OES delay in the PALs and the $\overline{\text{OESC}}$ -to-SC bus output-enable delay in the Am29C660C; this results in improved performance. This could not be done in an application where the CB bus and SC bus are tied together.

The Am29C660C must initialize the memory to a known state on Reset so that proper check bits are present for byte or word writes. On Reset, the data currently in the input latch is used to initialize the memory. If a known pattern must be written into memory, it may be done after the initialization in software.

System Data Bus Interface

The Am29C983s, U3 and U4, provide byte routing to and from the AT bus and internal D bus. The control signals for these devices are driven by the interface controller PALs, U10, U11 and U12.

Note that in the documentation for the PAL, the equations include a $\overline{\text{MEMW}}$ term. This term was included to prevent D-bus contention during a read without error cycle after T2.

Syndrome Latch (Syndrome Logic)

An Am29C823 9-bit latch, U5, stores the check bits when an error is detected by the EDC at timing tap T3. The contents of this latch can then be read by the microprocessor from the I/O channel in an interrupt routine. Decoding the syndrome latch bits reveals information about the error. Refer to the Am29C660C product specification for details. The error signal is qualified by MC1, RASI, and T3 so that the syndrome latch is only updated when errors occur during a Read or Write operation. This makes diagnostics easier and prevents glitches on $\overline{\text{EN}}$ of U5.

DRAM Array

The DRAM array consists of three banks of two blocks: the 32-bit data block and 7-bit check-bit block. The array is organized as three rows of 39 bits by 1 Mbit chips, or 117 components. Total user memory is 12 Mbytes. Note that the address space occupied on the AT bus is the 12 Mbytes immediately above the lowest megabyte of the 16 Mbytes available. By using a pulsed CAS signal to the chips, the data-in pins can be tied to the data-out pins on the DRAMs. The power pin on the 20-pin zip-pack memory chip is pin 15, the ground pin is 4, and no-connects are 8, 9, and 10.

JUMPER AND CONFIGURATION INFORMATION

A summary of options that can be modified by the user is given below. See the previous text for detailed information concerning these options.

- W1: selects the board I/O decode address (default = in)
- W2: IRQ_3 enable/disable to AT I/O channel (default = in)
- W3: determines the RM_2 input to the PAL DMTC (default = in)
- W4: determines CODEID₀ input to the EDC (default = in)

EDGE CONNECTOR PIN NAMES

Pin #	Signal Name	I/O
A1	—	—
A2	SD7	I/O
A3	SD6	I/O
A4	SD5	I/O
A5	SD4	I/O
A6	SD3	I/O
A7	SD2	I/O
A8	SD1	I/O
A9	SD0	I/O
A10	IOCHRDY	O
A11	AEN	I
A12	SA19	I
A13	SA18	I
A14	SA17	I
A15	SA16	I
A16	SA15	I
A17	SA14	I
A18	SA13	I
A19	SA12	I
A20	SA11	I
A21	SA10	I
A22	SA9	I
A23	SA8	I
A24	SA7	I
A25	SA6	I
A26	SA5	I
A27	SA4	I
A28	SA3	I
A29	SA2	I
A30	SA1	I
A31	SA0	I
C1	SBHE	I
C2	LA23	I
C3	LA22	I
C4	LA21	I
C5	LA20	I
C6	—	—
C7	—	—
C8	—	—
C9	MEMR	I
C10	MEMW	I
C11	SD8	I/O
C12	SD9	I/O
C13	SD10	I/O
C14	SD11	I/O
C15	SD12	I/O
C16	SD13	I/O
C17	SD14	I/O
C18	SD15	I/O

Pin #	Signal Name	I/O
B1	GND	—
B2	RESET	I
B3	+5 VDC	—
B4	—	—
B5	—	—
B6	—	—
B7	—	—
B8	—	—
B9	—	—
B10	GND	—
B11	—	—
B12	—	—
B13	LOW	I
B14	IOR	I
B15	—	—
B16	—	—
B17	—	—
B18	—	—
B19	REFRESH	I
B20	—	—
B21	—	—
B22	—	—
B23	—	—
B24	—	—
B25	IRQ3	O
B26	—	—
B27	—	—
B28	BALE	I
B29	+5 VDC	—
B30	—	—
B31	GND	—
D1	MEMCS16	O
D2	IOCS16	O
D3	—	—
D4	—	—
D5	—	—
D6	—	—
D7	—	—
D8	—	—
D9	—	—
D10	—	—
D11	—	—
D12	—	—
D13	—	—
D14	—	—
D15	—	—
D16	+5 VDC	—
D17	—	—
D18	GND	—

Note: A1-A31 and C1-C81 are on the component side, B1-B31 and D1-D18 are on the solder side.

PAL SOURCE CODE LISTINGS

The following application notes are guidelines to interfacing the Am29C668 with the microprocessor. They are paper designs only, and have not been built or tested. The assembler used is PLPL. Functional test vectors are used to verify these equations, but are not listed to conserve space.

```

DEVICE          IO_DECODER (P20L8)          "U7"

PIN             SA[11:0] = 1:11, 13 (INPUT Combinatorial)
               /IOW = 14 (INPUT Combinatorial)
               /IOR = 23 (INPUT Combinatorial)
               A320_220L = 16 (INPUT Combinatorial)
               AEN = 17 (INPUT Combinatorial)

               /SOE = 22 (OUTPUT Active_Low Combinatorial)
               /LEDIAG = 21 (OUTPUT Active_Low Combinatorial)
               /IOCS16 = 20 (OUTPUT Active_Low Combinatorial)
               /ENIOCS = 19 (OUTPUT Active_Low Combinatorial);

BEGIN

ENABLE (SOE, LEDIAG, ENIOCS);
ENABLE (AEN, A320_220L) = 0;
ENABLE (IOCS16) = ENIOCS;

CASE (SA[11:0])
BEGIN
  #B001000100000) BEGIN
    SOE = /A320_220L * IOR * /IOW * /AEN;
    IOCS16 = /A320_220L * /AEN * IOR;
    ENIOCS = /A320_220L * /AEN;
    END;

  #B001000100010) BEGIN
    LEDIAG = /A320_220L * IOW * /IOR * /AEN;
    IOCS16 = /A320_220L * /AEN * IOW;
    ENIOCS = /A320_220L * /AEN;
    END;

  #B001100100000) BEGIN
    SOE = A320_220L * IOR * /IOW * /AEN;
    IOCS16 = A320_220L * /AEN * IOR;
    ENIOCS = A320_220L * /AEN;
    END;

  #B001100100010) BEGIN
    LEDIAG = A320_220L * IOW * /IOR * /AEN;
    IOCS16 = A320_220L * /AEN * IOW;
    ENIOCS = A320_220L * /AEN;
    END;

END;

END.

```

```

DEVICE MEMORY_DECODER3 (P20L10) "U6"

PIN LA[23:21] = 1:3 (INPUT Combinatorial)
/REFRESH = 4 (INPUT Combinatorial)
/MEMR = 5 (INPUT Combinatorial)
/MEMW = 6 (INPUT Combinatorial)
BALE = 7 (INPUT Combinatorial)
/EOC = 8 (INPUT Combinatorial)
/INIT = 9 (INPUT Combinatorial)
T1 = 10 (INPUT Combinatorial)
T4 = 11 (INPUT Combinatorial)
/LATCHED_ERR = 13 (INPUT Combinatorial)

IOCHRDY = 23 (OUTPUT Active_Low Combinatorial)
/CYCREQ = 22 (OUTPUT Active_Low Combinatorial)
/MEMCS16 = 21 (OUTPUT Active_Low Combinatorial)
/ENMCS16 = 20 (OUTPUT Active_Low Combinatorial)
/RWRQ = 19 (OUTPUT Active_Low Combinatorial)
/RFRQ = 18 (OUTPUT Active_Low Combinatorial)
/FRH = 17 (OUTPUT Active_Low Combinatorial)
DMAC[23:21] = 16:14 (OUTPUT Active_Low Combinatorial);

DEFINE VALID_ADDRESS = LA[22] * /LA[21] + /LA[23] * LA[21] + LA[23] * /LA[22];

BEGIN

ENABLE (DMAC[23:21], CYCREQ, IOCHRDY, ENMCS16, RWRQ, RFRQ, FRH);
ENABLE (MEMCS16) = ENMCS16; ENABLE (IOCHRDY) = CYCREQ;

/DMAC[21] = /LA[21]; /DMAC[22] = /LA[22] * /LA[21] + LA[22] * LA[21];
/DMAC[23] = LA[23] * LA[22] + LA[23] * LA[21];

CYCREQ = VALID_ADDRESS * BALE * MEMW * /REFRESH * /INIT +
VALID_ADDRESS * BALE * MEMR * /REFRESH * /INIT +
/BALE * CYCREQ * (MEMW + MEMR) * /INIT;

MEMCS16 = VALID_ADDRESS * /REFRESH;
ENMCS16 = VALID_ADDRESS * /REFRESH;

/IOCHRDY = (MEMR + MEMW) * CYCREQ *
(MEMW * T1 * /LATCHED_ERR + T4 * (LATCHED_ERR + /MEMW));

RWRQ = VALID_ADDRESS * BALE * MEMW * /REFRESH * /INIT +
VALID_ADDRESS * BALE * MEMR * /REFRESH * /INIT +
RWRQ * (/EOC + CYCREQ) * /INIT;

RFRQ = REFRESH * FRH * /RWRQ * /EOC * /INIT + RFRQ * /EOC * /INIT;

FRH = / (REFRESH * /FRH + RFRQ * T4);

END.

```

```

DEVICE          ARBITER (P16L8) "U8"      MISC (P20L8)      DEVICE
PIN             T2 = 1 (INPUT Combinatorial)  T2 = 1 (INPUT Combinatorial)  PIN
                T8 = 2 (INPUT Combinatorial)  T8 = 2 (INPUT Combinatorial)
                T9 = 3 (INPUT Combinatorial)  T9 = 3 (INPUT Combinatorial)
                T10 = 4 (INPUT Combinatorial) T10 = 4 (INPUT Combinatorial)
                /RWRQ = 5 (INPUT Combinatorial) /RWRQ = 5 (INPUT Combinatorial)
                /RFRQ = 6 (INPUT Combinatorial) /RFRQ = 6 (INPUT Combinatorial)
                /LATCHED_ERR = 7 (INPUT Combinatorial) /LATCHED_ERR = 7 (INPUT Combinatorial)
                /INIT = 8 (INPUT Combinatorial) /INIT = 8 (INPUT Combinatorial)
                RM2 = 9 (INPUT Combinatorial) RM2 = 9 (INPUT Combinatorial)
                /MEMW = 11 (INPUT Combinatorial) /MEMW = 11 (INPUT Combinatorial)
                Done = 13 (INPUT Combinatorial) Done = 13 (INPUT Combinatorial)

                /EOC = 18 (OUTPUT Active_Low Combinatorial) /EOC = 18 (OUTPUT Active_Low Combinatorial)
                /RASI = 17 (OUTPUT Active_Low Combinatorial) /RASI = 17 (OUTPUT Active_Low Combinatorial)
                MC1 = 16 (OUTPUT Active_Low Combinatorial) MC1 = 16 (OUTPUT Active_Low Combinatorial)
                MC0 = 15 (OUTPUT Active_Low Combinatorial) MC0 = 15 (OUTPUT Active_Low Combinatorial)
                /R_WL = 14 (OUTPUT Active_Low Combinatorial) /R_WL = 14 (OUTPUT Active_Low Combinatorial)

BEGIN
ENABLE (RASI,EOC,MC0,MC1,R_WL);
ENABLE (Done) = 0;

EOC = INIT * T8 +
    RFRQ * (/RM2 * T8 + RM2 * T10) +
    RWRQ * /RFRQ *
    (/R_WL * T10 + R_WL * /LATCHED_ERR * T8 +
    R_WL * LATCHED_ERR * T10) +
    EOC * T8;

RASI = INIT * T2 +
    /INIT * /RFRQ * /RWRQ +
    RFRQ * /RM2 * T2 * /MC0 * /MC1 +
    RFRQ * RM2 * T9 * MC0 * /MC1 +
    RWRQ * /RFRQ * R_WL * (/LATCHED_ERR * T2 +
    LATCHED_ERR * T9) +
    RWRQ * /RFRQ * /R_WL * T9;

/MC0 = INIT + RFRQ * RM2 * (/T9 + /RASI);

/MC1 = /(RFRQ * /RM2 * (/T2 + /RASI) +
    RFRQ * RM2 * (/T9 + /RASI) + INIT * DONE);

R_WL = MEMW + R_WL * /EOC;

END.

```

```

DEVICE          MISC  (P20L8)          " U 1 2 "          DEVICE

PIN            T2 = 1 (INPUT Combinatorial)
               T3 = 2 (INPUT Combinatorial)
               T6 = 3 (INPUT Combinatorial)
               T7 = 4 (INPUT Combinatorial)
               T9 = 5 (INPUT Combinatorial)
               /CYCREQ = 6 (INPUT Combinatorial)
               /LATCHED_ERR = 7 (INPUT Combinatorial)
               /INIT = 8 (INPUT Combinatorial)
               RM2 = 9 (INPUT Combinatorial)
               R_WL = 10 (INPUT Combinatorial)
               RASI = 11 (INPUT Combinatorial)
               /RWRQ = 13 (INPUT Combinatorial)
               /RFRQ = 23 (INPUT Combinatorial)
               /LATCHED_MERR = 14 (INPUT Combinatorial)

               /WE[2:0] = 22:20 (OUTPUT Active_Low Combinatorial)
               /LEO_GENL = 19 (OUTPUT Active_Low Combinatorial)
               /LEI = 18 (OUTPUT Active_Low Combinatorial)
               LEB = 17 (OUTPUT Active_Low Combinatorial)
               /S_AND_NOT_LEB = 16 (OUTPUT Active_Low Combinatorial);

BEGIN

ENABLE (LEB,LEI,LEO_GENL,WE[2:0],S_AND_NOT_LEB);

/LEB = RWRQ * /RFRQ * R_WL * RASI *
      (/T2 * /LATCHED_ERR + /T3 * LATCHED_ERR);

LEI = INIT + /INIT * /RFRQ * /RWRQ +
      RWRQ * /RASI + RWRQ * T7 * /LEO_GENL * /R_WL +
      RWRQ * T7 * /LEO_GENL * R_WL * LATCHED_ERR +
      RWRQ * T7 * R_WL * /LATCHED_ERR +
      RFRQ * /RM2 + RFRQ * RM2 * /RASI + RFRQ * RM2 * T7 * /LEO_GENL;

LEO_GENL = /INIT * (RFRQ * RM2 +
      RWRQ * /RFRQ * (/R_WL + R_WL * LATCHED_ERR)) * T3 * /T9;

WE[2] = INIT + RFRQ * RM2 * T6 * /T9 * /LATCHED_MERR +
      RWRQ * /RFRQ * (R_WL * LATCHED_ERR + /R_WL) * T6 * /T9 * /LATCHED_MERR;

WE[1] = INIT + RFRQ * RM2 * T6 * /T9 * /LATCHED_MERR +
      RWRQ * /RFRQ * (R_WL * LATCHED_ERR + /R_WL) * T6 * /T9 * /LATCHED_MERR;

WE[0] = INIT + RFRQ * RM2 * T6 * /T9 * /LATCHED_MERR +
      RWRQ * /RFRQ * (R_WL * LATCHED_ERR + /R_WL) * T6 * /T9 * /LATCHED_MERR;

S_AND_NOT_LEB = /INIT * RASI * /T2 * (RWRQ * /RFRQ + RFRQ) +
      RWRQ * /RFRQ * R_WL * RASI * (/T2 * /LATCHED_ERR + /T3 * LATCHED_ERR);

END.

```

```

DEVICE          Output_Enable  (P20L8)          "U10"

PIN
    T2 = 1 (INPUT Combinatorial)
    T7 = 2 (INPUT Combinatorial)
    T8 = 3 (INPUT Combinatorial)
    T10 = 4 (INPUT Combinatorial)
    SA0 = 5 (INPUT Combinatorial)
    SA1 = 6 (INPUT Combinatorial)
    /SBHE = 7 (INPUT Combinatorial)
    /CYCREQ = 8 (INPUT Combinatorial)
    /LATCHED_ERR = 9 (INPUT Combinatorial)
    /INIT = 10 (INPUT Combinatorial)
    RM2 = 11 (INPUT Combinatorial)
    R_WL = 13 (INPUT Combinatorial)
    /RWRQ = 14 (INPUT Combinatorial)
    /RFRQ = 23 (INPUT Combinatorial)

    /OEBH = 15 (OUTPUT Active_Low Combinatorial)
    /OEBL = 16 (OUTPUT Active_Low Combinatorial)
    /OEHL = 17 (OUTPUT Active_Low Combinatorial)
    /OEHO = 18 (OUTPUT Active_Low Combinatorial)
    /OEL1 = 19 (OUTPUT Active_Low Combinatorial)
    /OEL0 = 20 (OUTPUT Active_Low Combinatorial)
    /OES = 21 (OUTPUT Active_Low Combinatorial);

DEFINE          B_WL = /SBHE + SA0;

BEGIN

ENABLE (OEBH, OEBL, OEHL, OEHO, OEL1, OEL0);

OEBH = /INIT * RWRQ * /RFRQ * R_WL * T7 * CYCREQ * SA1;

OEBL = /INIT * RWRQ * /RFRQ * R_WL * T7 * CYCREQ * /SA1;

OEL0 = INIT + (RFRQ * RM2 * T2 * /T10) +
    RWRQ * /RFRQ * /R_WL * (B_WL * (SA1 + SA0) + /B_WL * SA1) * T2 * /T10 +
    RWRQ * /RFRQ * R_WL * LATCHED_ERR * T2 * /T10;

OEL1 = INIT + (RFRQ * RM2 * T2 * /T10) +
    RWRQ * /RFRQ * /R_WL * (B_WL * (SA1 + /SA0) + /B_WL * SA1) * T2 * /T10 +
    RWRQ * /RFRQ * R_WL * LATCHED_ERR * T2 * /T10;

OEHO = INIT + (RFRQ * RM2 * T2 * /T10) +
    RWRQ * /RFRQ * /R_WL * (B_WL * (/SA1 + SA0) + /B_WL * /SA1) * T2 * /T10 +
    RWRQ * /RFRQ * R_WL * LATCHED_ERR * T2 * /T10;

OEHL = INIT + (RFRQ * RM2 * T2 * /T10) +
    RWRQ * /RFRQ * /R_WL * (B_WL * (/SA1 + /SA0) + /B_WL * /SA1) * T2 * /T10 +
    RWRQ * /RFRQ * R_WL * LATCHED_ERR * T2 * /T10;

END.

```



```

DEVICE          INTERFACE (P20L8)          "U11"          DEVICE
PIN

/OE0 = 1 (INPUT Combinatorial)
/OE1 = 2 (INPUT Combinatorial)
/OE2 = 3 (INPUT Combinatorial)
/OE3 = 4 (INPUT Combinatorial)
S_AND_NOT_LEB = 5 (INPUT Combinatorial)
/MEMW = 6 (INPUT Combinatorial)
LEO_GENL = 7 (INPUT Combinatorial)
LEDIAG = 8 (INPUT Combinatorial)
T2 = 9 (INPUT Combinatorial)
CAS = 10 (INPUT Combinatorial)
TS = 11 (INPUT Combinatorial)
/ERR = 13 (INPUT Combinatorial)
RASI = 23 (INPUT Combinatorial)
MC1 = 14 (INPUT Combinatorial)
T3 = 16 (INPUT Combinatorial)

LG = 22 (OUTPUT Active_Low Combinatorial)
/OE_BYTE3 = 21 (OUTPUT Active_Low Combinatorial)
/OE_BYTE2 = 20 (OUTPUT Active_Low Combinatorial)
/OE_BYTE1 = 19 (OUTPUT Active_Low Combinatorial)
/OE_BYTE0 = 18 (OUTPUT Active_Low Combinatorial)
CASI = 17 (OUTPUT Active_Low Combinatorial)
/SYN_EN = 15 (OUTPUT Active_Low Combinatorial);

BEGIN

ENABLE (OE_BYTE0,OE_BYTE1,OE_BYTE2,OE_BYTE3,LG,CASI,SYN_EN);
ENABLE (T3) = 0;

OE_BYTE0 = (/OE0 * S_AND_NOT_LEB * MEMW) + LEDIAG;
OE_BYTE1 = (/OE1 * S_AND_NOT_LEB * MEMW) + LEDIAG;
OE_BYTE2 = /OE2 * S_AND_NOT_LEB * MEMW;
OE_BYTE3 = /OE3 * S_AND_NOT_LEB * MEMW;

/LG = MEMW + LEDIAG;
/CASI = (/T2 * CAS) + (TS * CAS);

SYN_EN = ERR * MC1 * RASI * /T3;

END.

```

DEVICE	SAMPLE (P20RA10)	"U9"	DEVICE
PIN	<pre> /PRE_LOAD = 1 (CONTROL) /ERR = 2 (INPUT Combinatorial) /INT2 = 3 (INPUT Combinatorial) /MERR = 4 (INPUT Combinatorial) /INT4 = 5 (INPUT Combinatorial) /SOE = 6 (INPUT Combinatorial) /TC = 7 (INPUT Combinatorial) /SYS_RESET = 8 (INPUT Combinatorial) /EOC = 9 (INPUT Combinatorial) /T8 = 10 (INPUT Combinatorial) /OE = 13 (CONTROL) /LATCHED_ERR = 23 (OUTPUT Active_Low Registered) /INTR3 = 22 (OUTPUT Active_Low Registered) /INIT = 21 (OUTPUT Active_Low Registered) /Done = 20 (OUTPUT Active_Low Registered) /Counter[0:2] = 19:17 (OUTPUT Active_Low Registered) /LATCHED_MERR = 16 (OUTPUT Active_Low Registered); </pre>		PIN
BEGIN			
ENABLE (LATCHED_ERR, INTR3, Done, Counter[0:2], INIT, LATCHED_MERR);			
LATCHED_ERR = ERR; CLOCK_PT (LATCHED_ERR) = INT2;			
PRESET (LATCHED_ERR) = EOC;			
INTR3 = MERR + INTR3; CLOCK_PT (INTR3) = INT4;			
RESET (INTR3) = SOE;			
LATCHED_MERR = MERR; CLOCK_PT (LATCHED_MERR) = INT4;			
RESET (LATCHED_MERR) = EOC;			
CLOCK_PT (INIT) = SYS_RESET; RESET (INIT) = TC * EOC; INIT = 1;			
RESET (Counter[2:0]) = /INIT; CLOCK_PT (Counter[2:0]) = T8;			
IF (/Done = 1) THEN			
CASE (Counter[2:0]) BEGIN			
0) Counter[2:0] = 1;			
1) Counter[2:0] = 2;			
2) Counter[2:0] = 3;			
3) Counter[2:0] = 4;			
4) Counter[2:0] = 5;			
5) Counter[2:0] = 6;			
6) Counter[2:0] = 7;			
7) Counter[2:0] = 0;			
END;			
Done = Counter[2] * Counter[1] * Counter[0] + Done * INIT;			
RESET (Done) = /INIT; CLOCK_PT (Done) = T8;			
END.			

```

DEVICE          TIMER (P20RA10)  "U2"          "U13" (P20RA10)  DEVICE
PIN             /PRE_LOAD = 1 (CONTROL)          /PRE_LOAD = 1 (CONTROL)          PIN
INT1 = 2 (INPUT Combinatorial)          /ERR = 2 (INPUT Combinatorial)
INT2 = 3 (INPUT Combinatorial)          /INT3 = 3 (INPUT Combinatorial)
INT3 = 4 (INPUT Combinatorial)          /INT4 = 4 (INPUT Combinatorial)
INT4 = 5 (INPUT Combinatorial)          /INT5 = 5 (INPUT Combinatorial)
INT6 = 6 (INPUT Combinatorial)          /INT6 = 6 (INPUT Combinatorial)
INT7 = 7 (INPUT Combinatorial)          /INT7 = 7 (INPUT Combinatorial)
INT8 = 8 (INPUT Combinatorial)          /INT8 = 8 (INPUT Combinatorial)
INT9 = 9 (INPUT Combinatorial)          /INT9 = 9 (INPUT Combinatorial)
INT10 = 10 (INPUT Combinatorial)          /INT10 = 10 (INPUT Combinatorial)
/EOC = 11 (INPUT Combinatorial)          /EOC = 11 (INPUT Combinatorial)
/OE = 13 (CONTROL)          /OE = 13 (CONTROL)

T1 = 23 (OUTPUT Active_Low Registered)
T2 = 22 (OUTPUT Active_Low Registered)
T3 = 21 (OUTPUT Active_Low Registered)
T4 = 20 (OUTPUT Active_Low Registered)
T6 = 19 (OUTPUT Active_Low Registered)
T7 = 18 (OUTPUT Active_Low Registered)
T8 = 17 (OUTPUT Active_Low Registered)
T9 = 16 (OUTPUT Active_Low Registered)
T10 = 15 (OUTPUT Active_Low Registered);

BEGIN
ENABLE (T1, T2, T3, T4, T6, T7, T8, T9, T10);

/T1 = 1;      CLOCK_PT (T1) = INT1;      PRESET (T1) = EOC;
/T2 = 1;      CLOCK_PT (T2) = INT2;      PRESET (T2) = EOC;
/T3 = 1;      CLOCK_PT (T3) = INT3;      PRESET (T3) = EOC;
/T4 = 1;      CLOCK_PT (T4) = INT4;      PRESET (T4) = EOC;
/T6 = 1;      CLOCK_PT (T6) = INT6;      PRESET (T6) = EOC;
/T7 = 1;      CLOCK_PT (T7) = INT7;      PRESET (T7) = EOC;
/T8 = 1;      CLOCK_PT (T8) = INT8;      PRESET (T8) = EOC;
/T9 = 1;      CLOCK_PT (T9) = INT9;      PRESET (T9) = EOC;
/T10 = 1;     CLOCK_PT (T10) = INT10;     PRESET (T10) = EOC;

END.

```

DELAY LINE TAP CALCULATIONS

Derivation of the tap outputs is included here. The calculated time is adjusted to the nearest tap of the delay line equal to or greater than the calculated time.

	Calculated Time (ns)
MSEL - RASI to MUX SELECT	
t_{RAH} (DRAM) min	15.0
t_{SKEW} (Qn to \overline{RASn}) 29C668 max	6.0
Total	21.0
\overline{CAS} - RASI to \overline{CAS}	
MSEL	21.0
t_{SKEW} (\overline{CASn} to Qn) 29C668 max	-2.0
t_{ASC} DRAM min	0.0
$-t_{PD}$ (\overline{CAS} to CASI) 20L8B min	-6.0
Total	13.0
INT7 - Data Valid to 29C660C	
t_{PD} (RASI to \overline{RASn}) 29C668 max	29.0
t_{ACC} DRAM max	120.0
t_{SU} (Data In) 29C660C min	3.0
$-t_{PD}$ (INT7 to T7) 20RA10 min	-7.0
Total	145.0
INT2 - \overline{ERROR} from 29C660C	
t_{PD} (RASI to \overline{RASn}) 29C668 max	29.0
t_{ACC} DRAM max	120.0
t_{PD} (Data In to \overline{ERROR}) 29C660C max	16.0
t_{SU} (\overline{ERROR}) 20RA10 min	13.0
Total	178.0
INT3 - Corrected Data from 29C660C	
t_{PD} (RASI to \overline{RASn}) 29C668 max	29.0
t_{ACC} DRAM max	120.0
t_{PD} (Data In to Data Out) 29C660C max	24.0
$-t_{PD}$ (INT3 to T3) 20RA10 min	-7.0
Total	166.0
INT4 - \overline{MERR} from 29C660C and $\overline{IOCHRDY}$ for R/M/W	
t_{PD} (RASI to \overline{RASn}) 29C668 max	29.0
t_{ACC} DRAM max	120.0
t_{PD} (Data In to MULT ERROR) 29C660C max	20.0
t_{SU} (\overline{MERR}) 20RA10 min	13.0
Total	182.0

Calculated Time
(ns)INT 1 - $\overline{IOCHRDY}$ for Read without Error

t_{PD} (RASI to \overline{RASn}) 29C668 max	29.0
t_{ACC} DRAM max	120.0
t_{PD} (Data In to Data Out) 29C660C max	24.0
t_{PD} (Data Out to System Data) 29C983 max	14.0
$-t_{PD}$ (T1 to $\overline{IOCHRDY}$) 20L10B min	-6.0
$-t_{PD}$ (INT1 to T1) 20RA10 min	-7.0
Total	174.0

INT 6 - Corrected Data and Check Bits (R/M/W)

INT3	166.0
t_{SKEW} (T6 to T3) 20RA10 max	0.5
t_{PD} (T3 to LEO \overline{GEN}) 20L8B max	15.0
t_{PD} (LEO \overline{GEN} to SCn) 29C660C max	18.0
t_{DS} DRAM min	0.0
Total	199.5

TS - Pulsed \overline{CAS}

INT6	199.5
t_{PD} (INT6 to T6) 20RA10 max	20.0
t_{PD} (T6 to \overline{WE}) 20L8B max	15.0
$-t_{PD}$ (TS to CASI) 20L8B min	-6.0
$-t_{PD}$ (CASI to \overline{CASn}) 29C668 min	-16.0
t_{WCS} DRAM	0.0
Total	228.5

INT9 - End of \overline{WE} and RASI (R/M/W)

t_s	222.5
t_{PD} (TS to CASI) 20L8B max	15.0
t_{PD} (CASI to \overline{CASn}) 29C668 max	16.0
t_{WCH} (\overline{WE} Pulse Width) DRAM min	25.0
$-t_{PD}$ (T9 to \overline{WE}) 20L8B min	-6.0
$-t_{PD}$ (INT9 to T9) 20RA10 min	-7.0
Total	261.5

INT8 - End of Read without Error

INT2	178.0
t_{RP} RAM min	90.0
Total	268.0

INT10 - End of R/M/W Cycle

INT9	261.5
t_{RP} RAM min	90.0
Total	351.5

Signal	Should Be (ns)	Is (ns)	Note
MSEL	21.0	30.0	
CAS	22.0	30.0	= MSEL - 8
INT7	145.0	150.0	
INT2	178.0	180.0	
INT3	166.0	170.0	
INT4	182.0	190.0	
INT1	174.0	180.0	
INT6	203.5	210.0	= INT3 + 33.5
t _s	239.0	240.0	= INT6 + 29
INT8	270.0	270.0	= INT2 + 90
INT9	273.0	280.0	= TS + 33
INT10	370.0	370.0	= INT9 + 90

Notes:

1. Taps dependent or related to other taps are indicated with a comment in the "explanation" column.
2. Timing figures are based on Am29C660C data.

PARTS LIST

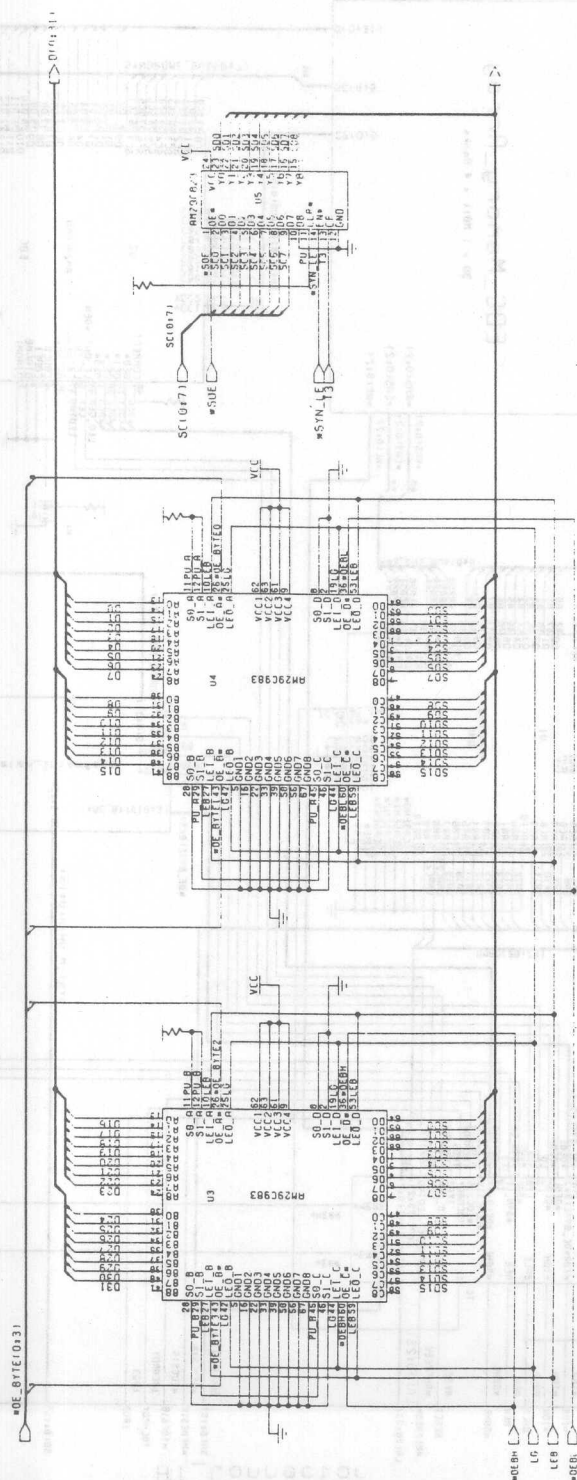
Unit #	Device	Description
U1	Am29C668	Configurable Dynamic Memory Controller
U2	Am29C660C	32-Bit Cascadable Error Detection and Correction Circuit
U3	Am29C983	Multiple Bus Exchange
U4	Am29C983	Multiple Bus Exchange
U5	Am29C823	Syndrome Latch for diagnostic use
U6	Am20L10B	Memory Decoder
U7	Am20L8B	I/O Decoder
U8	Am16L80	Control Logic
U9	Am20RA10	Asynchronous PAL
U10	Am20L8B	Interface Controller
U11	Am20L8B	Interface Controller
U12	Am20L8B	Interface Controller
U13	Am20RA10	Timing Tap Outputs
U14	Delay Line	System Timing Generation
U15	Delay Line	System Timing Generation
U16	Delay Line	System Timing Generation
U17	Delay Line	System Timing Generation

Description	Quantity per Board
CAPACITOR, 22 uF	7
CAPACITOR, 1.0 uF	1
CAPACITOR, 0.33 uF	76
CAPACITOR, 0.1 uF	27
CAPACITOR, 0.01 uF	1
RESISTOR PACK, 10 PIN SIP, 1 k	1
RESISTORS, 39 Ω	3
DRAM, ZIP PACK, 1 M x 1	117
AM29C660C	1
AM29C668	1
AM29C983	2
AM29C823	1
AM16L8D	1
AM20L8B	4
AM20L10B	1
AM20RA10-20	2
DELAY LINE, 10 ns, DIP-14	4
SOCKET, 14 PIN DIP	4
SOCKET, 20 PIN DIP	1
SOCKET, 24 PIN DIP	9
SOCKET, 68 PIN PGA/PLCC CONV.	4

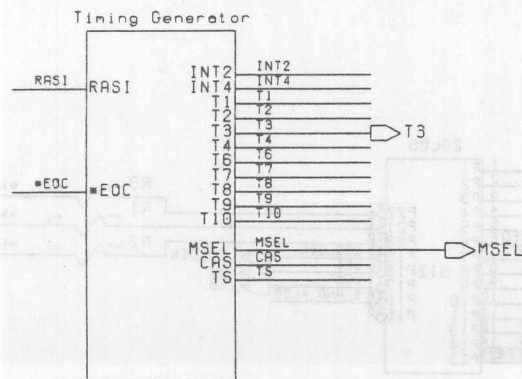
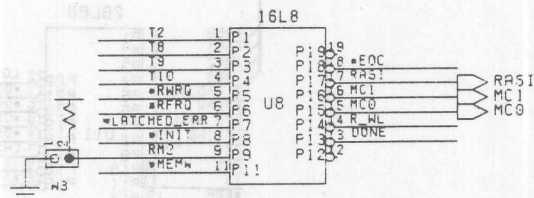
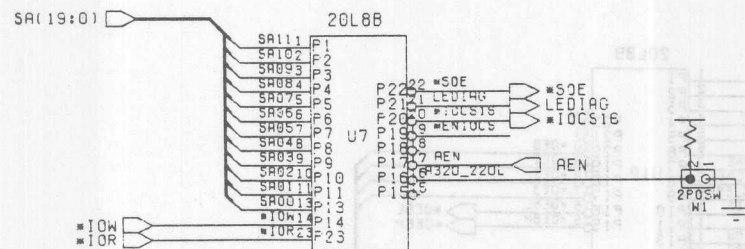
SCHEMATICS

Detailed schematics follow.

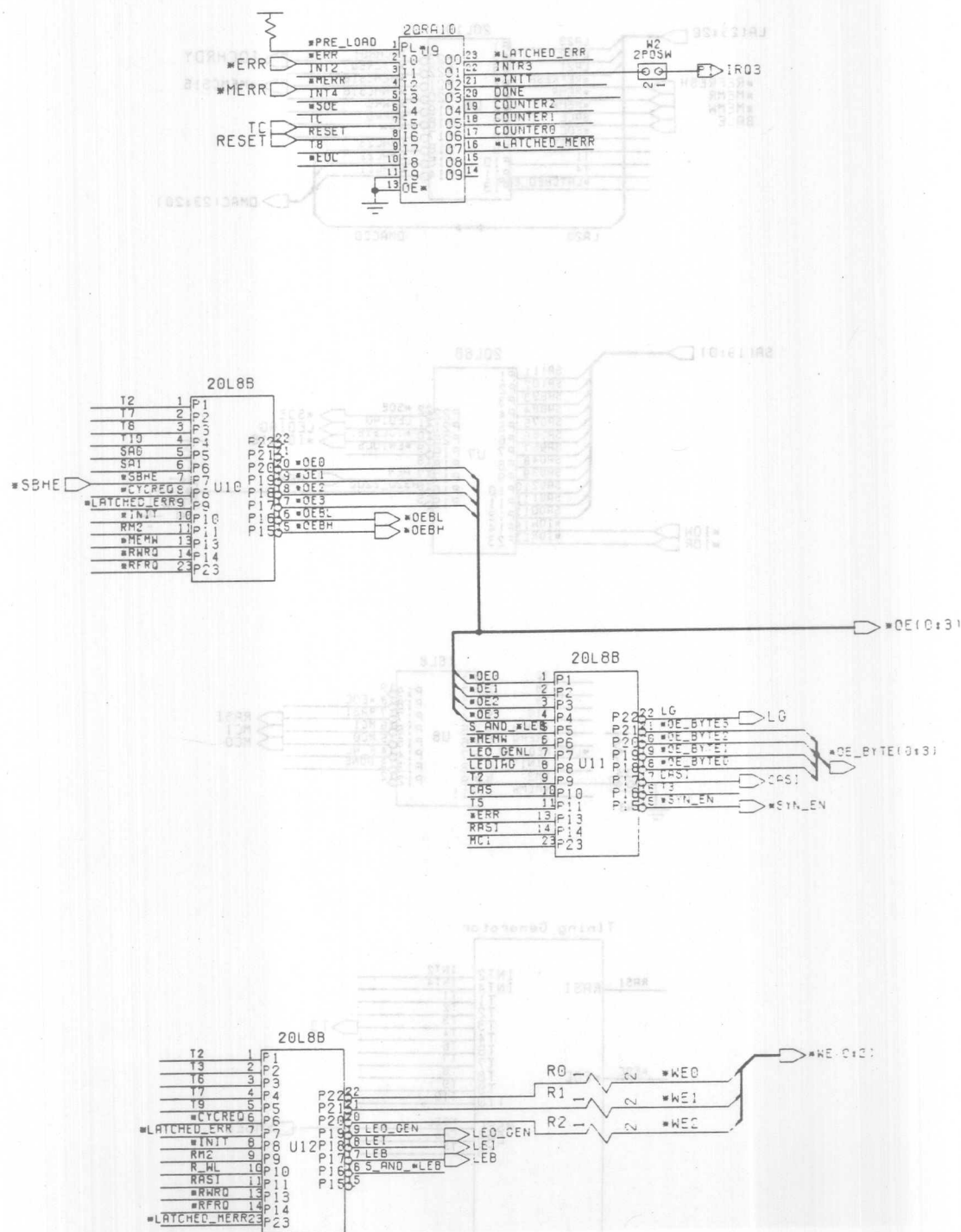




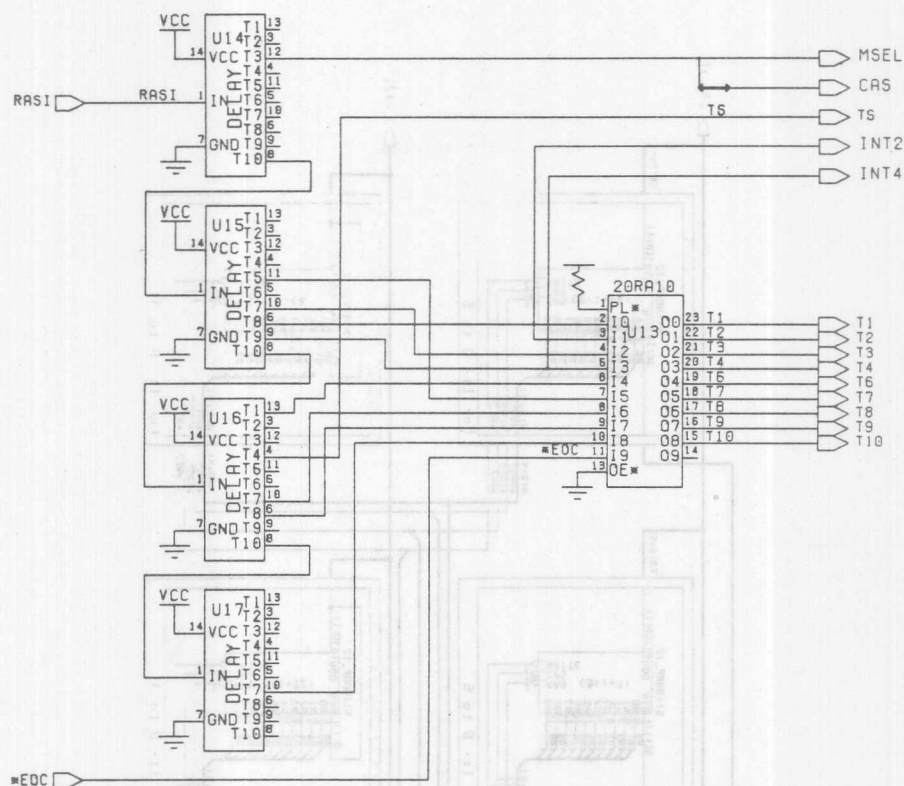
System Data Interface



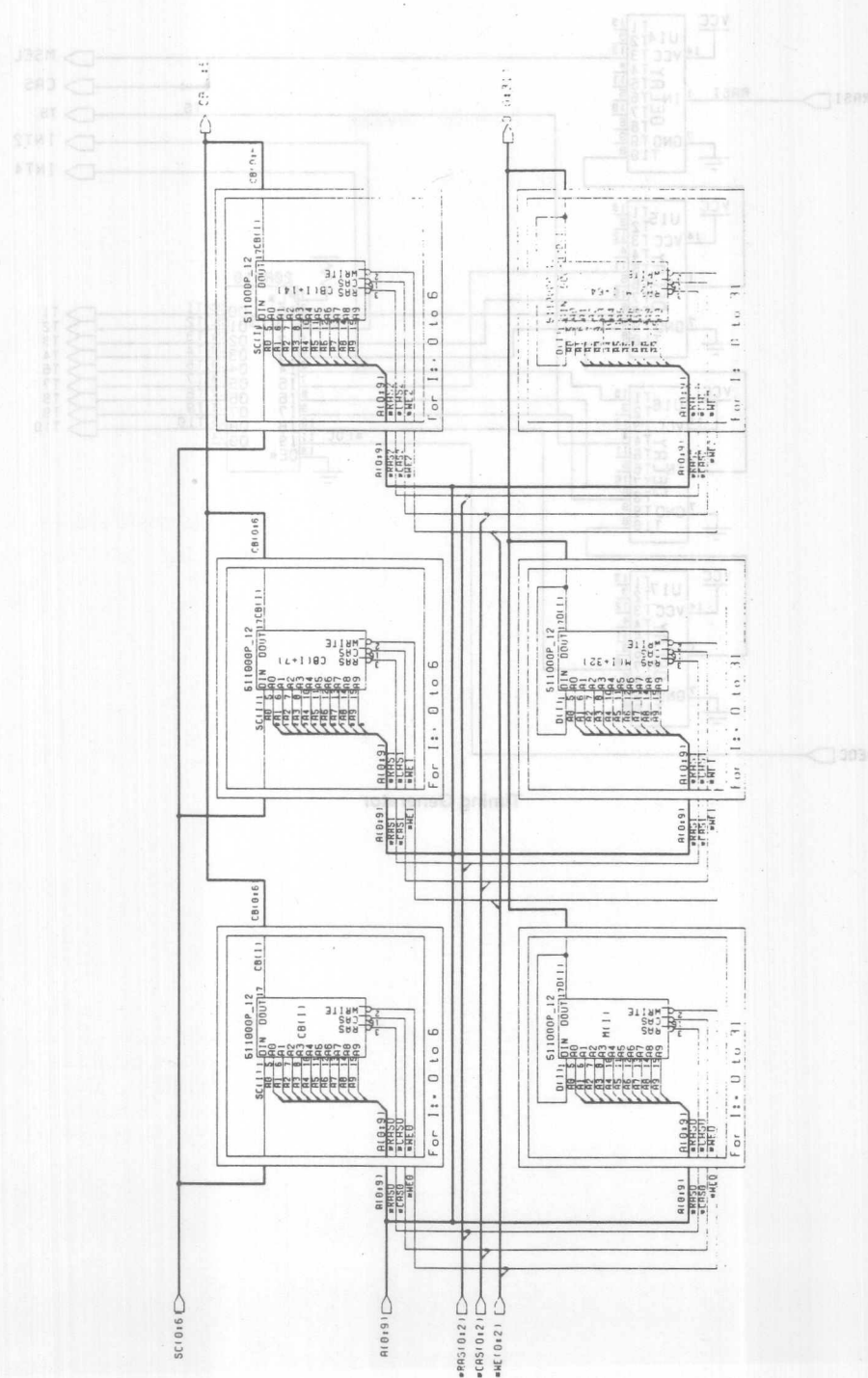
4-21



Interface Controller Logic



Timing Generator



Memory Configuration

IBM PS/2* 12-Mbyte Memory Board with Error Detection and Correction (EDC)



INTRODUCTION

This Micro-Channel-compatible evaluation board demonstrates the capabilities of the Am29C668 4M Configurable Dynamic Memory Controller (CDMC) and the Am29C660C Error Detection and Correction Circuit (EDC). As newer systems and software demand much larger memories, it becomes increasingly more important to protect the memory from soft errors, thereby increasing system reliability. Soft errors occur when a single bit is complemented due to noise, alpha particles or some other event. The most common error is a single-bit error where one bit of a memory word is incorrect. The Am29C660 EDC detects and corrects all single-bit errors and detects all double and some multiple-bit errors. When a word is accessed, it is checked for errors; if an error is found, the corrected data is written back. This board also performs memory "scrubbing," which is the detection and correction of single-bit errors during refresh to maintain the integrity of seldom-accessed memory locations. Scrubbing the memory prevents accumulation of single-bit errors. Double-bit errors result when two single-bit errors occur in the same word. Since the probability of this happening is quite low, scrubbing memory prevents most double-bit errors.

The Am29C668 CDMC is capable of controlling large memories, up-to-four banks of 4-Mbit DRAMS, and driving the \overline{RAS}_n , \overline{CAS}_n and address lines without external drivers or damping resistors. It automatically generates the addresses needed for normal row refresh and refresh with scrubbing. The CDMC also has many features not utilized in this design, but appropriate for other systems, e.g., this design does not require reconfiguration of the CDMC through a simple I/O interface (see CDMC discussion, page 5).

Distinctive Characteristics

- 12 Mbytes of dynamic RAM (1M x 1-bit packages). 12 DRAM modules and 9 zip packages are used for maximum component density.
- Am29C668 4M Configurable Dynamic Memory Controller/Driver.
- Am29C660C high-speed 32-bit Error Detection and Correction Circuit corrects all single-bit errors, detects all double and some multiple-bit errors.
- One Wait State at 16 MHz with 120-ns DRAMs. Zero Wait States at 16 MHz with 70-ns DRAMs. One Wait State at 20 MHz with 85-ns DRAMs. Supports both basic transfer cycles and matched memory cycles.

- Supports memory scrubbing during refresh for improved reliability.
- The ability to relocate memory and I/O space through the Programmable Option Select (POS) registers. All options are software configurable through the POS registers.
- Dynamic Memory Timing Controller implemented using Programmable Array Logic (PAL[®]) devices and delay lines.
- Am29C688 used in Am29368-compatible mode with logic to reconfigure the Am29C668.
- 32-bit internal data bus with 7-bit check bit and 7-bit syndrome bus.
- Syndrome latch for diagnostic and test purposes.
- Direct interface with PS/2 Model 70 and 80 systems.

A BRIEF OVERVIEW OF THE MICRO-CHANNEL ARCHITECTURE

The Micro-Channel bus used in IBM PS/2 systems provides for three different add-in cards: 16-bit, 16-bit with auxiliary video extension and 32-bit. This board is designed for 32-bit systems and fits only in the current IBM PS/2 Models 70 and 80 systems. The Micro Channel supports two types of bus accesses: Matched Memory Cycles and Basic Transfer Cycles. The Basic Transfer Cycle is supported by all PS/2 models. It permits at least 200-ns minimum cycle time with wait states of at least 100 ns. A card designed to support this type of access can be used in any of the PS/2 models. Matched Memory Cycles are only supported in 80386 machines, currently Models 70 and 80; cycle time is dependent upon the processor cycle time. Each access is a minimum of three processor cycles; however, additional wait states may be added. Cards designed to support this type of access cannot be used in all machines. This design supports Matched Memory Accesses, because it provides for the highest performance. Table 1 shows the number of wait states for specific memory access times and processor speeds for Read accesses. If only Basic Transfer Cycles are used, all Read accesses require one wait state (300 ns cycle time) to complete. Table 2 shows the read-access time (Status Valid to Read Data Valid) for different speed DRAMs.

Table 1. Number of Wait States for Processor Speed and Memory Access Time

Memory Access Time-ns	Processor Clock 16 MHz	20 MHz
120	1	2
100	1	2
85	1	1
80	1	1
70	0	1

Table 2. Read Access Times For Different DRAMs from Status Active

DRAM Access Time-ns	Memory Board Access Time-ns
120	188
100	168
85	153
80	148
70	138

DETAILED DESCRIPTION

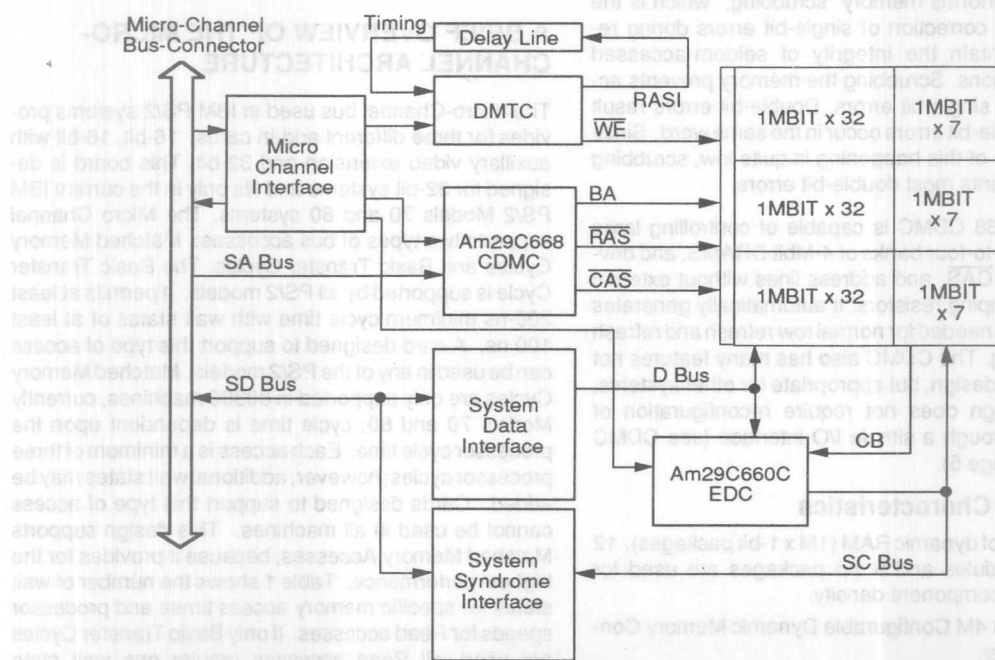
The primary data paths and functional elements are shown in Figure 1. The following discussion describes each section of the block diagram in detail. Components not appearing in the block diagram but existing on the schematic are also discussed.

Edge Card Connectors

This board can only be used in a 32-bit Micro Channel backplane. Interrupt Request $\overline{IRQ3}$ is used by the board to signal the detection of a multiple-bit error to the system processor. The interrupt can be disabled by writing a zero to bit 0 of POS Register 104. All signals used from the edge connector are listed on pages 8 and 9.

Dynamic Memory Timing Control (DMTC)

The timing controller for this board was implemented using PAL devices and delay lines for increased flexibility and performance. The following subsections describe the signals and their function.



11587-001A

Figure 1. Block Diagram

I/O Channel Ready Logic

I/O Channel Ready $\overline{\text{IOCHRDY}}$ is used to signal the processor that valid data is ready during Reads and to signal completions of Writes. When $\overline{\text{IOCHRDY}}$ is pulled High, the processor inserts wait states until $\overline{\text{IOCHRDY}}$ is asserted. This signal is very important since it must be deasserted before Command Signal $\overline{\text{CMD}}$ is asserted by the system. If this does not happen, the system does not insert wait states and data is lost or corrupted.

There are two different types of extended cycles during a basic transfer cycle: synchronous and asynchronous. Synchronous extended cycles insert only one wait state. $\overline{\text{IOCHRDY}}$ is asserted within 30 ns of $\overline{\text{CMD}}$ going active. During asynchronous extended cycles, $\overline{\text{IOCHRDY}}$ is asserted 60 ns at most, before Read data is valid. To insert a wait state during a matched memory cycle, $\overline{\text{IOCHRDY}}$ is asserted 45 ns, at most, before Read data is valid. As the processor speed increases, this time proportionally decreases, i. e., a 20-MHz machine has 33 ns before Read data is valid. For this board to work in faster machines, $\overline{\text{IOCHRDY}}$ is asserted when the data is valid. In this manner, the board functions properly in any system with only a minor speed penalty in slower machines. For basic transfer cycles, the board uses synchronous extended cycles to maximize the memory bandwidth.

If a cycle has not completed, a wait state must be inserted regardless of the next type of access. The signal $\overline{\text{BBar}}$ and Busy are used to handle this logic. When the system initiates a memory access, signaled by $\overline{\text{CMD}}$ going active, $\overline{\text{BBar}}$ is active. $\overline{\text{BBar}}$ remains active until the end of the board memory cycle, signaled by End of Cycle $\overline{\text{EOC}}$. When the system ends its memory access by deactivating $\overline{\text{CMD}}$, Busy goes active and remains active until the end of the board memory cycle. If any other access to the board is attempted while Busy is High, $\overline{\text{IOCHRDY}}$ is deasserted and wait states inserted until the board's memory cycle terminates.

RAS_i, Mode Selection and End of Cycle

The Row Address Strobe Input RAS_i, Mode Control MC_n, and End of Cycle $\overline{\text{EOC}}$ signals are generated by U4. RAS_i is used to initiate the timing sequence and to signal the Am29C668 to generate the RAS_n signals to the appropriate bank of memory. Two different sets of mode signals are generated, AC[2:0] and MC[1:0]. The AC[2:0] signals are used for internal control within the DMTC. An encoding scheme for the memory state was selected to minimize inputs to the PAL devices. If a fully decoded scheme were used, six signals instead of three would be required. There is no speed penalty since the memory state must be latched for the duration of the memory cycle and the encoding and latching are all performed by one PAL. Table 3 shows the decoding of AC[2:0].

Table 3. AC[2:0] Decoding

AC[2:0]	Mode
000	No Operation (Idle)
001	Long Write (32-bits)
010	Write
011	Not Allowed
100	Read
101	Refresh without Scrubbing
110	Refresh with Scrubbing
111	Initialize

MC₁ and MC₀ control the type of memory access for the CDMC. Table 4 shows the decoding of MC₁ and MC₀.

Table 4. MC₁ and MC₀ Decoding

MC ₁	MC ₀	Mode
0	0	Refresh without scrubbing
0	1	Refresh with scrubbing or initialize
1	0	Read/Write mode
1	1	Reset Refresh Counter

The $\overline{\text{EOC}}$ signal is generated to signify the end of any memory cycle. This signal also resets AC[2:0]. When AC[2:0] = 000, the End of Timing (EOT) becomes active and resets the timing-tap output to ensure that there can be no glitch on RAS_i. If $\overline{\text{EOC}}$ resets both AC[2:0] and the timing taps, the timing taps may be reset before AC[2:0] is reset. The RAS_i logic goes High until AC[2:0] is reset, resulting in a glitch on RAS_i and consequently on RAS_n to the DRAMs.

Latched Error, Initialization and Interrupt Signals

There are two different cycle lengths: a short cycle used by Read without Error, Refresh, Long Write (32-bits) and Initialize and a long cycle used by Read with Error, Read/Modify/Write and Scrubbing. All cycle lengths except Read cycles are known at the beginning. Because of this, careful attention must be paid to the timing and logic used during Read cycles. The EDC generates the $\overline{\text{ERROR}}$ signal when a single or multiple-bit error is detected during a Read, Read/Modify/Write or Scrubbing cycle. $\overline{\text{LErr}}$ is used by the rest of the board to determine if a Read cycle is long or short. PAL20RA10, U5, samples $\overline{\text{ERROR}}$ at Timing Tap T2 and, if $\overline{\text{ERROR}}$ is false, signal $\overline{\text{LErr}}$ is deasserted. $\overline{\text{LErr}}$ is preset by $\overline{\text{EOC}}$. This logic assumes that every Read cycle is a long cycle unless $\overline{\text{ERROR}}$ is false at T2. This assures correct and concise logic implementation. If $\overline{\text{LErr}}$ were conditionally asserted instead of deasserted, much more complicated logic would be required, since another signal is needed to indicate when $\overline{\text{LErr}}$ is valid.

Device U5 also latches the Initialize $\overline{\text{INIT}}$ signal, which is generated by the Board Enable $\overline{\text{BDENBL}}$ going active. $\overline{\text{INIT}}$ remains active until Terminal Count TC is received from the Am29C668 indicating that all the memory locations have been initialized. Counter[0:3] counts the wake-up cycles. When eight wake-up cycles are completed, $\overline{\text{DONE}}$ is asserted signaling the DMTC that it can begin initializing memory. At the end of a wake-up cycle, Counter[0:3] is incremented. Counter[0:3] is initialized to 0 and when the count reaches 7, $\overline{\text{DONE}}$ is asserted and the counter is inhibited. $\overline{\text{DONE}}$ remains active until $\overline{\text{INIT}}$ is deactivated.

Memory-Board Interrupt $\overline{\text{INTR}}$ signals that a multiple error has occurred at time T4. $\overline{\text{INTR}}$ goes to Interrupt Request $\overline{\text{IRQ3}}$ on the backplane and is cleared by an access to the syndrome latch signal SynLE . This signal can be disabled by writing a zero to bit 0 of POS register 104. Registering $\overline{\text{INTR3}}$ would not be required in systems that support bus retry. The Channel Check signal ChCk can also be generated by writing a one to bit 3 of POS Register 104. Device U5 generates SetChCk when a multiple error is detected and ChCk is enabled. This signal is normally disabled.

Pulsed CASI, Write Enable and Miscellaneous Logic Functions

The Column Address Strobe Input CASI is a pulsed CAS line used when connecting the data-in lines to the data-out lines on the DRAMs. The Interface Controller PAL U11 generates this signal from the registered timing-tap signals from the Micro Channel Interface EPB2001, U13. Figure 2 shows how the pulsed-CASI signal is produced. This only applies during Read/Modify/Write and Refresh-with-Scrubbing cycles. Note that at time A in the diagram, the DRAM outputs are three-stated so that the Am29C660 can drive the data bus.

Write Enable $\overline{\text{WE}}$ is used to write the valid data into the memory. One $\overline{\text{WE}}$ signal is used per bank to drive the high capacitive load. The total delay must be calculated since the load capacitance is greater than the load specified in the data sheet ($\text{CL} = 50 \text{ pF}$). The load capacitance of the DRAMs is $(4 \times 60) + (3 \times 5) = 255 \text{ pF}$, and the internal resistance of the PAL is assumed to be 4Ω during High-to-Low transitions. The maximum High-to-Low transition time is calculated from 4.0 V to 0.8 V. The final output voltage is 0.5 V; therefore, the maximum High-to-Low transition time is:

$$V_{\text{OUT}} = (4.0 - 0.5) \exp\left(\frac{-t}{4 \Omega \times 255 \text{ pF}}\right) + 0.5$$

$$t(\text{High to Low}) = (4 \Omega)(255 \text{ pF})(-1) \ln\left(\frac{0.8 - 0.5}{4.0 - 0.5}\right) \approx 2.5 \text{ ns}$$

This is added to the worst-case t_{pd} (15 ns) to get the worst-case delay. The minimum High-to-Low transition is calculated from 2.4 to 0.8 V with a final voltage of 0.3 V:

$$V_{\text{OUT}} = (2.4 - 0.3) \exp\left(\frac{-t}{50 \Omega \times 255 \text{ pF}}\right) + 0.3$$

$$t(\text{High to Low}) = (4 \Omega)(255 \text{ pF})(-1) \ln\left(\frac{0.8 - 0.3}{2.4 - 0.3}\right) \approx 1.5 \text{ ns}$$

The minimum Low-to-High transition is from 0.8 V to 2.4 V with a final voltage of 4.0 V. The internal resistance of the PAL is approximately 50Ω :

$$V_{\text{OUT}} = (0.8 - 4.0) \exp\left(\frac{-t}{50 \Omega \times 255 \text{ pF}}\right) + 4.0$$

$$t(\text{Low to High}) = (50 \Omega)(255 \text{ pF})(-1) \ln\left(\frac{2.4 - 4.0}{0.8 - 4.0}\right) \approx 9 \text{ ns}$$

Miscellaneous logic functions are performed by U6, a combinatorial PAL device. Latch Enable Output or Generate LEO_GenL is a dual-purpose signal; when active High, it enables the output latches of the

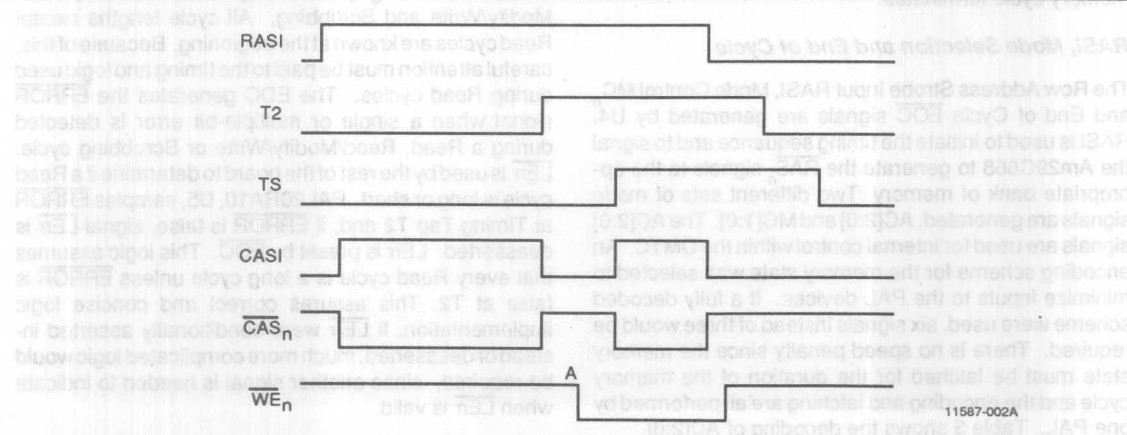


Figure 2. Idealized Timing Diagram for Pulsed CASI Signal

Am29C660; when it is active Low, the Am29C660 generates check bits for the data in the input latch. The Latch Enable Input signal LEI controls the latching of data into the Am29C660. When LEI is High, the input latch is transparent; when LEI is Low, data is latched. The Latch Enable Bus signal LEB controls the data latch from the internal data bus to the system bus. When LEB is High, the latch is transparent. S_AND_LEB is used to condition the output enables from the system bus.

Delay Lines

Delay lines D1 through D4 provide the timing reference signals; RASI initiates the timing sequence. See page 18-19 for the tap-timing calculations of the timing configuration currently installed on the board. The board is designed for 120-ns DRAMs. According to the board timing requirements, each timing signal must be reset at the beginning of each cycle. A PAL20RA10, U9, is used to register the timing taps. Since it has separate clocks for each of its 10 output registers, this PAL saves board space over a discrete-logic implementation. The outputs are reset by \overline{EOC} going active. Registering the timing taps allows shorter cycle times, since it is not necessary to wait for the delay line to clear.

Interface Control

PAL U7 generates Output Enable System Data $\overline{OE_SD}[0:3]$, Output Enable EDC $\overline{OE_EDC}[0:3]$ and LG signals. Output Enable System Data controls the gating of data from the system bus to the memory. These signals control the flow of data during Writes to memory and to the EDC diagnostic register. For diagnostic Writes, the lower word (16 bits) from the data bus is input to the Am29C660. For Writes, the data is controlled by the $\overline{BE}[0:3]$ signals generated by the system board. Output Enable EDC determines which data bytes the EDC supplies during Write and Read/Modify/Write cycles. The EDC supplies the unaltered bytes during a Write and provides the corrected bytes during a Read/Modify/Write cycle. LEY controls the input data latches of the bus transceivers. When the signal is active, the data latch is transparent; when it is inactive, the system data is latched.

PAL U8 generates $\overline{OE_BD}[0:3]$, Syndrome Output Enable, Syndrome Latch Enable and Diagnostic Latch Enable. Syndrome Output Enable \overline{SYNOE} , an active-Low signal, enables the Am29C823 to drive the data from the syndrome latch onto the system data bus. Syndrome Latch Enable $\overline{SYN_LE}$ is an active-Low signal that latches the syndrome bits when an error is detected. $\overline{OE_BD}[0:3]$ drives the $\overline{OE_C}$ and $\overline{OE_D}$ inputs of the transceivers that provide the system data-bus interface. These signals gate the data lines to and from the various byte-wide data bits of the internal data bus on the board, the D bus. Note from the PAL equations that the gating

signals are conditioned by S_AND_LEB to ensure proper latching of the data from the DRAMs. LEY enables the latches on Writes; \overline{LEB} enables the latches on Reads.

Configurable Dynamic Memory Controller

The Am29C668 Configurable Dynamic Memory Controller U1 supplies the DRAM array with multiplexed address, \overline{RAS}_n and \overline{CAS}_n signals. Timing inputs to this device are provided by the delay lines registered by U9. The Am29C668 is used in the Am29C368-compatible mode and can be reconfigured by writing data to the configuration registers.

During Initialization, the Am29C668 generates initialization cycles until the entire memory is written with data and check bits. When the initialization is complete, TC is asserted High signaling the DMTC that the initialization is complete. \overline{RAS} -only Refresh is used when no memory scrubbing is selected. Note: AC10 and AR10 are not connected since 1-Mbit DRAMs are used; 1-Mbit DRAMs use only 20 address bits.

Error Detection and Correction Circuit

The high-speed Am29C660C EDC U2 is used in the correct-always mode, i.e., data is always corrected before it is output to the bus. The fly-by mode, where the processor is interrupted when errors occur, cannot be used with PS/2 systems because the 80286/386 microprocessors do not support bus retry. In systems that support bus retry, data is read from the board as soon as it is accessed from memory. This saves 24 ns t_{pd} Data In to Data Correct for the C-speed part, during memory Reads. Memory Write times are not changed.

This device generates check bits during a Write and verifies the data and check bits during a Read. Separate error \overline{ERROR} and multiple-bit error $\overline{MULT_ERROR}$ signals are output. If $\overline{MULT_ERROR}$ is asserted, then IRQ3 is active if enabled by IRQOE. Code ID generates input signal CODE ID0 to U2. If Code ID is active, a 32-bit slice is selected (the chip may operate in 32- or 64-bit mode); if Code ID is not active, the chip operates in internal-control mode. Using the internal-control mode, the user can access the diagnostic registers in the Am29C660C and more easily debug and test the board. See the Am29C660 data sheet for further details.

The Am29C660 internal diagnostic latch is available from the I/O channel. Data is written to the diagnostic latch through an I/O address specified through the POS registers. Bits 2 and 3 of POS Register 102 select four different addresses for the diagnostic latch. A 16-bit data word is written to the register to configure the part. Consult the Am29C660 data sheet for further information.

The $\overline{\text{OESC}}$ pin on the EDC is grounded to eliminate the OESC-to-SC bus output-enable delay in the Am29C660 with a resulting improvement in performance. This could not be done in an application where the CB bus and SC bus are tied together.

The Am29C660 must initialize the memory to a known state on reset. On $\overline{\text{BdEnb}}$ going active, the data currently in the input latch is used to initialize the memory. If a known pattern must be written into memory, it may be done after the hardware initialization in software.

System Data Bus Interface

The 74F543s, U15 to U18, provide data latching between the system bus and internal D bus. The control signals for these devices are driven by the interface controller PAL devices, U7 and U8. Note that in the documentation for the PAL, the equations include a MemWr term, included to prevent D-bus contention during a Read-without-Error cycle after T2. The latches in the 74F543 are used to latch the Write data and free the bus. If simple transceivers are used, the data on the bus must be held until the Write data set-up time for the DRAMs is satisfied, adding to the access time. By using the 74F543s, 100 ns are saved during Read/Modify/Write cycles by latching the data and releasing the bus after the Read access is completed. This is not a problem on Long Write cycles since the data can be written directly to memory.

Syndrome Register (Syndrome Logic)

An Am29C823 register U12 stores the syndrome bits when an error is detected by the Am29C660C at timing tap T3. The contents of this register can then be read by the microprocessor from the I/O channel in an interrupt routine. Decoding the syndrome register bits reveals information about the error that occurred (see Am29C660 data sheet). The error signal is qualified by MemRd and MemWr so that the syndrome latch can only be updated when errors occur during a Read or Write operation. $\overline{\text{INIT}}$ is connected to the $\overline{\text{CLR}}$ of the Am29C823 to clear the latch on power-up and system reset.

DRAM Array

The DRAM array consists of two rows of two blocks: the 32-bit data block and 7-bit check-bit block. It is organized as three rows of 39 bits by 1 Mbit devices, or 117 components. Total user memory is 12 Mbytes. By using a pulsed $\overline{\text{CAS}}$ signal to the chips, the data-in pins can be tied to the data-out pins on the DRAMs. This facilitates routing on the PC board by minimizing the number of traces to the DRAM array. Four 9-bit memory modules and three zip packages are used per memory bank. The first four bits in each module are data bits. The last bit in each module and the three zip packages are used to store the check bits. The ninth bit of each module has

separate data-in and data-out lines, while the rest of the module has common data-in and data-out lines. The check bits require separate data-in and data-out lines, since $\overline{\text{OES}}$ is tied Low to minimize the delay from check-bit generation to write back. This design uses 120-ns DRAMs to minimize cost. Using faster memories lowers the access times and reduces the number of wait states needed (See Tables 1 and 2).

Micro-Channel Bus Interface

Devices U13 and U14 perform most of the Micro-Channel interface. U14 is a comparator that compares the upper eight address bits with the eight bits set in POS register 105. If the upper bytes match and MADE24 is inactive, AddressValid32 is active. U13 is a user-configurable Adapter Interface device, the EPB2001, designed specifically for the PS/2 Micro Channel. It decodes the lower 24 bits of the address, AddressValid32 , $\overline{\text{MIO}}$, $\overline{\text{SO}}$ and $\overline{\text{S1}}$ and signal-valid memory accesses and I/O accesses. $\overline{\text{CdDS16}}$ and $\overline{\text{CdDS32}}$ are also generated by this device. $\overline{\text{CdDS16}}$ is generated during an access to the syndrome latch, Am29C660's diagnostic register, Am29C668 configuration register or to memory. $\overline{\text{CdDS32}}$ is only asserted during memory accesses. Both $\overline{\text{CdDS16}}$ and $\overline{\text{CdDS32}}$ are generated during memory accesses to indicate that the memory supports both 16- and 32-bit transfers.

Unit 13 also contains all the POS registers. Bits 0 to 2 of POS register 104 are output on POS I/O 0 to 2 and are used to drive the DMTC inputs IRQEN , RM and CodeID . By writing the appropriate values to these bits, the board is configured (Figure 3). MemRd and MemWr generated by the DMTC logic could have been generated from the Micro-Channel Interface device, U13. This was not done because it is faster to begin Read cycles when status becomes valid, rather than wait for U13 to generate them. Generating these signals via the DMTC logic saves 55 ns on basic transfer cycles and 82 ns on matched memory cycles at 16 MHz.

POS CONFIGURATION INFORMATION

Figure 3 shows the mapping of control bits in the POS registers. The use of POS registers eliminates the need for jumpers and helps the user to easily resolve conflicts in memory and I/O mapping. This design provides maximum flexibility when reconfiguring the board via this interface.

POS-register 104 is used to configure the modes of the board. If Bit 0 is zero, interrupts from the board are disabled; a one enables the interrupts. Bit 1 determines the type of refresh, one for scrubbing and zero for non-scrubbing. Bit 2 determines the mode of the Am29C660C. If Bit 2 is zero, the board is in normal

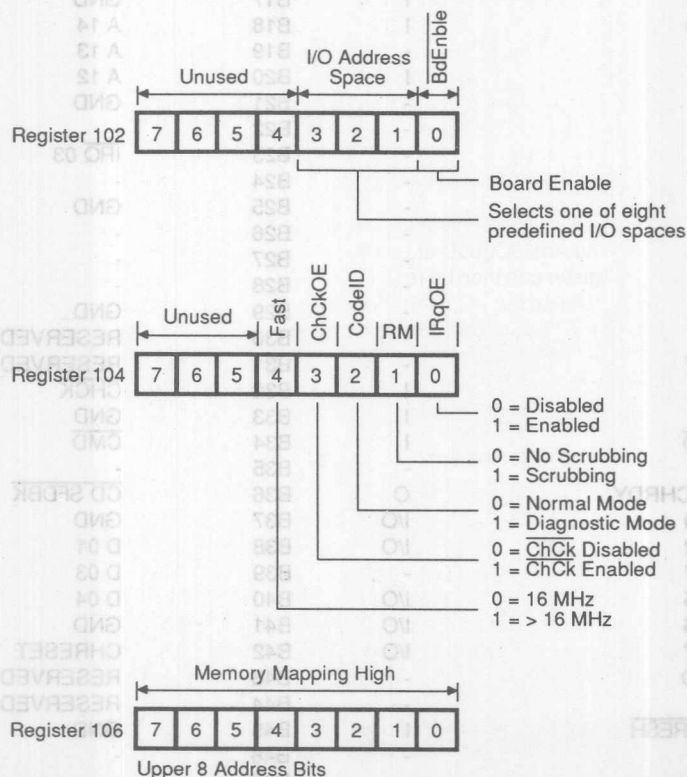
operating mode; if one, the board is running diagnostics and using the values in the diagnostic register to control the operation of the Am29C660C. Bit 4 is the fast bit for indicating the processor speed, zero for 16-MHz systems and one for faster systems. This bit is used in generating the wait states for the board.

POS-register 102 is used to determine the I/O address space. Bits 0:2 select one of eight address spaces for the CDMC, Syndrome and Diagnostic Latch. The total I/O space for the board is 4 Kbytes. Since an I/O-mapped scheme is used to reconfigure the Am29C668, it occupies the lower 2 Kbytes I/O address space. The lower 11 bits of the address, bits A 00 to A 10, are the configuration data used by the Am29C668. The CDMC could have been mapped to a single I/O address with the configuration data written on the data bus, but this would require a multiplexer to select between the data and address bus, which adds extra board space, control logic and delay to the system. Using this I/O-mapped scheme, the Micro-Channel interface can drive $\overline{\text{Cd DS}} 16$ and $\overline{\text{Cd DS}} 32$, saving additional logic. The Syndrome Latch I/O address

is byte 0 and the Diagnostic Latch I/O address is byte 1 of the upper 2 Kbytes of the I/O space.

POS-register 106 contains the upper eight bits of the memory board. These eight bits are compared with the upper eight bits on the system bus to determine if the access is the same address space as the memory.

POS-registers 100 and 101 contain the board ID for identifying the card during setup. POS-register 102 bit 0 is used as the board-enable signal. This bit is reset by ChReset or by the processor writing a zero to this bit during a card-setup cycle. While this bit is zero, the board does not respond to any access. This bit can only be reset by the processor during card-setup cycles and cannot be set during normal I/O Writes to this register. POS-register-105 bit 7 is the channel-check flag, the state of this bit is output on the bus through the ChCk pin. This bit is set by ChReset or by writing a one to the location. The bit is reset by asserting SetChk or by writing a zero to the location.



11587-003A

Figure 3. POS Registers Bit Map

EDGE CONNECTOR PIN NAMES

Pin #	Signal Name	I/O	Pin #	Signal Name	I/O
AM4	-	-	BM4	GND	-
AM3	MMC CMD	I	BM3	-	-
AM2	GND	-	BM2	MMCR	O
AM1	MMC	I	BM1	-	-
A1	CD SETUP	I	B1	-	-
A2	MADE 24	-	B2	-	-
A3	GND	-	B3	GND	-
A4	A 11	I	B4	-	-
A5	A 10	I	B5	GND	-
A6	A 09	I	B6	A 23	I
A7	+5 V	-	B7	A 22	I
A8	A 08	I	B8	A 21	I
A9	A 07	I	B9	GND	-
A10	A 06	I	B10	A 20	I
A11	+5 V	-	B11	A 19	I
A12	A 05	I	B12	A 18	I
A13	A 04	I	B13	GND	-
A14	A 03	I	B14	A 17	I
A15	+5 V	-	B15	A 16	I
A16	A 02	I	B16	A 15	I
A17	A 01	I	B17	GND	-
A18	A 00	I	B18	A 14	I
A19	-	-	B19	A 13	I
A20	ADL	I	B20	A 12	I
A21	-	-	B21	GND	-
A22	-	-	B22	-	-
A23	-	-	B23	IRQ 03	O
A24	-	-	B24	-	-
A25	-	-	B25	GND	-
A26	-	-	B26	-	-
A27	-	-	B27	-	-
A28	-	-	B28	-	-
A29	-	-	B29	GND	-
A30	-	-	B30	RESERVED	-
A31	+5 V	-	B31	RESERVED	-
A32	$\overline{S0}$	I	B32	CHCK	O
A33	$\overline{S1}$	I	B33	GND	-
A34	M/IO	I	B34	CMD	I
A35	-	-	B35	-	-
A36	CD CHRDY	O	B36	CD SFDBK	O
A37	D 00	I/O	B37	GND	-
A38	D 02	I/O	B38	D 01	I/O
A39	+5 V	-	B39	D 03	I/O
A40	D 05	I/O	B40	D 04	I/O
A41	D 06	I/O	B41	GND	-
A42	D 07	I/O	B42	CHRESET	I
A43	GND	-	B43	RESERVED	-
A44	-	-	B44	RESERVED	-
A45	REFRESH	I	B45	GND	-
A46	-	-	B46	-	-
A47	-	-	B47	-	-
A48	+5 V	-	B48	D 08	I/O
A49	D 10	I/O	B49	D 09	I/O
A50	D 11	I/O	B50	GND	-

Pin #	Signal Name	I/O	Pin #	Signal Name	I/O
A51	D 13	I/O	B51	D 12	I/O
A52	-	-	B52	D 14	I/O
A53	RESERVED	-	B53	D 15	I/O
A54	-	-	B54	GND	-
A55	$\overline{\text{Cd DS 16}}$	O	B55	-	-
A56	+5 V	-	B56	-	-
A57	-	-	B57	-	-
A58	-	-	B58	GND	-
A59	RESERVED	-	B59	RESERVED	-
A60	RESERVED	-	B60	RESERVED	-
A61	GND	-	B61	RESERVED	-
A62	RESERVED	-	B62	RESERVED	-
A63	RESERVED	-	B63	GND	-
A64	RESERVED	-	B64	D 16	I/O
A65	-	-	B65	D 17	I/O
A66	D 19	I/O	B66	D 18	I/O
A67	D 20	I/O	B67	GND	-
A68	D 21	I/O	B68	D 22	I/O
A69	+5 V	-	B69	D 23	I/O
A70	D 24	I/O	B70	RESERVED	-
A71	D 25	I/O	B71	GND	-
A72	D 26	I/O	B72	D 27	I/O
A73	+5 V	-	B73	D 28	I/O
A74	D 30	I/O	B74	D 29	I/O
A75	D 31	I/O	B75	GND	-
A76	RESERVED	-	B76	BE 0	I
A77	-	-	B77	BE 1	I
A78	$\overline{\text{BE 3}}$	I	B78	BE 2	I
A79	-	-	B79	GND	-
A80	$\overline{\text{Cd DS 32}}$	O	B80	-	-
A81	-	-	B81	A 24	I
A82	A 26	I	B82	A 25	I
A83	A 27	I	B83	GND	-
A84	A 28	I	B84	A 29	I
A85	+5 V	-	B85	A 30	I
A86	RESERVED	-	B86	A 31	I
A87	RESERVED	-	B87	GND	-
A88	RESERVED	-	B88	RESERVED	-
A89	GND	-	B89	RESERVED	-

Note: Side A is the component side, Side B is on the solder side.

- I = Signal Input to the board.
- O = Signal Output from the board.
- I/O = Signal Input to and Output from the board.
- = Not Applicable.

PAL SOURCE CODE LISTINGS

The following application notes are guidelines to interfacing the Am29C668 with the microprocessor. They are paper designs, and have not been built or tested. The assembler used was PLPL. Functional test vectors were used to verify these equations, but are not listed to conserve space.

"August 9, 1988

32-Bit Error Detection and Correction Board for the Micro Channel."

DEVICE Channel_Ready (P22V10)

"U3"

```

PIN Fast = 1 (INPUT Combinatorial)
    AC[2:0] = 2:4 (INPUT Combinatorial)
    LongWord = 5 (INPUT Combinatorial)
    /MemReq = 6 (INPUT Combinatorial)
    /Cmd = 7 (INPUT Combinatorial)
    /CdDS16 = 8 (INPUT Combinatorial)
    /MMCmd = 9 (INPUT Combinatorial)
    /MMC = 10 (INPUT Combinatorial)
    EoC = 11 (INPUT Combinatorial)
    /BdEnbl = 13 (INPUT Combinatorial)
    T1 = 14 (INPUT Combinatorial)

    ALE = 23 (OUTPUT Active_High Combinatorial)
    /IOChRdy = 22 (OUTPUT Active_Low Combinatorial)
    /MMCr = 21 (OUTPUT Active_Low Combinatorial)
    /MemWr = 20 (OUTPUT Active_Low Combinatorial)
    /MemRd = 19 (OUTPUT Active_Low Combinatorial)
    BBar = 18 (OUTPUT Active_High Combinatorial)
    Busy = 17 (OUTPUT Active_High Combinatorial)
    Delayed = 16 (OUTPUT Active_High Combinatorial);

```

```

DEFINE Read = /AC[0] * /AC[1] * AC[2],
    Write = /AC[0] * AC[1] * /AC[2],
    Long_Write = AC[0] * /AC[1] * /AC[2];

```

BEGIN

ENABLE (Busy, LongWord, BBar); ENABLE (T1, BdEnbl) = 0;

ENABLE (IOChRdy) = BdEnbl;

```

IOChRdy = CdDS16 * Delayed + Fast * /T1 +
    /Fast * /Delayed * /Cmd + /Fast * /Delayed * /MMCmd + /Fast * Delayed * /T1;
Delayed = Busy * (S0 + S1) + Delayed * /Eoc * MemReq;

```

```

BBar = Cmd * MemReq + BBar * /EoC;
Busy = BBar * /Cmd + Busy * /EoC;

```

```

ALE = /Cmd * /MMCmd * /Busy;
MemRd = (Cmd + MMCmd) * Read;
MemWr = (Cmd + MMCmd) * (Write + Long_Write);

```

MMCr = MMC * MemReq * /Delayed;

END.

32-Bit Error Detection and Correction Board for the Micro Channel.[™]

```

DEVICE          RASI      (P22V10)      " U 4 "

PIN      /Cmd = 1 (INPUT Combinatorial)
        LongWord = 2 (INPUT Combinatorial)
        /MemReq = 3 (INPUT Combinatorial)
        /S0 = 4 (INPUT Combinatorial)
        /S1 = 5 (INPUT Combinatorial)
        /FR = 6 (INPUT Combinatorial)
        RM = 7 (INPUT Combinatorial)
        /Init = 8 (INPUT Combinatorial)
        T2 = 9 (INPUT Combinatorial)
        T8 = 10 (INPUT Combinatorial)
        T9 = 11 (INPUT Combinatorial)
        T10 = 13 (INPUT Combinatorial)
        /Latched_Err = 14 (INPUT Combinatorial)
        Done = 15 (INPUT Combinatorial)
        Busy = 16 (INPUT Combinatorial)

        AC[0:2] = 21:23 (OUTPUT Active_High Combinatorial)
        MC1 = 20 (OUTPUT Active_High Combinatorial)
        MC0 = 19 (OUTPUT Active_High Combinatorial)
        RASI = 18 (OUTPUT Active_High Combinatorial)
        /EoC = 17 (OUTPUT Active_Low Combinatorial);

DEFINE Read = /AC[0] * /AC[1] * AC[2],
        Write = /AC[0] * AC[1] * /AC[2],
        Long_Write = AC[0] * /AC[1] * /AC[2];

BEGIN

ENABLE (RASI,EoC,MC0,MC1,EoT,AC[0:2]);
ENABLE (Done,Busy,Latched_Err) = 0;

AC[0] = (LongWord * MemWr) * /Cmd + (Refresh * /RM + Init) * /Busy + AC[0] * /EoC;
AC[1] = (/LongWord * MemWr) * /Cmd + (Refresh * RM + Init) * /Busy + AC[1] * /EoC;
AC[2] = (MemRd + Refresh + Init) * /Busy + AC[1] * /EoC;

EoC = Init * T8 + Refresh * /RM * T8 + Refresh * RM * T10 + Write * T10 +
        Long_Write * T8 + Read * (/Latched_Err * T8 + Latched_Err * T10) +
        EoC * T8;

RASI = (Init * /T2 + Refresh * /RM * /T2 * /MC0 * /MC1 + Refresh * RM * /T9 *
        MC0 * /MC1 + (MemRd + Read) * (/Latched_Err * /T2 + Latched_Err * /T9) +
        (MemWr + Write) * /T9 + MemWr * LongWord * /T2) * /Busy;

MC0 = Init + Refresh * RM * (/T9 + /RASI) + DMCSel;
MC1 = /(Refresh * /RM * (/T2 + /RASI) + Refresh * RM * (/T9 + /RASI) + Init * Done);

END.

```

"August 9, 1988

32-Bit Error Detection and Correction Board for the Micro Channel."

DEVICE

SAMPLE (P20RA10)

"U5"

```

PIN    /PRE_LOAD = 1 (CONTROL)
      INT2 = 2 (INPUT Combinatorial)
      INT4 = 3 (INPUT Combinatorial)
      ChCkOE = 4 (INPUT Combinatorial)
      /EoC = 5 (INPUT Combinatorial)
      /SynSel = 6 (INPUT Combinatorial)
      TC = 7 (INPUT Combinatorial)
      /BdEnbl = 8 (INPUT Combinatorial)
      /Err = 9 (INPUT Combinatorial)
      /MErr = 10 (INPUT Combinatorial)
      IRqOE = 11 (INPUT Combinatorial)
      /OE = 13 (CONTROL)

      /LErr = 23 (OUTPUT Active_Low Registered)
      /Intr = 22 (OUTPUT Active_Low Registered)
      /INIT = 21 (OUTPUT Active_Low Registered)
      /Done = 20 (OUTPUT Active_Low Registered)
      /Counter[0:2] = 19:17 (OUTPUT Active_Low Registered)
      /LMErr = 16 (OUTPUT Active_Low Registered)
      /SetChCk = 15 (OUTPUT Active_Low Combinatorial);

BEGIN

ENABLE (LErr, Intr, Done, Counter[0:2], INIT, LMErr);

LErr = Err;
CLOCK_PT (LErr) = INT2;
PRESET (LErr) = EoC;

Intr = MErr + Intr;
CLOCK_PT (Intr) = INT4;
RESET (Intr) = SynSel;
ENABLE (Intr) = IRqOE * BdEnbl;
LMErr = MErr;
CLOCK_PT (LMErr) = INT4;
RESET (LMErr) = EoC;

CLOCK_PT (INIT) = BdEnbl;
RESET (INIT) = TC * EoC;
INIT = 1;

RESET (Counter[2:0]) = /INIT;
CLOCK_PT (Counter[2:0]) = EoC;
IF (/Done = 1)
THEN CASE (Counter[2:0])
BEGIN
  0) Counter[2:0] = 1;
  1) Counter[2:0] = 2;
  2) Counter[2:0] = 3;
  3) Counter[2:0] = 4;
  4) Counter[2:0] = 5;
  5) Counter[2:0] = 6;
  6) Counter[2:0] = 7;
  7) Counter[2:0] = 0;

```

```

END;
Done = Counter[2] * Counter[1] * Counter[0] + Done * INIT;
CLOCK_PT(Done) = EoC;
RESET(Done) = /INIT;
SetChCk = ChCkOE * LMErr;
END.

```

32-Bit Error Detection and Correction Board for the Micro Channel.

DEVICE MISC (P22P10)

```

PIN AC[0:2] = 3:1 (INPUT Combinatorial)
RASI = 4 (INPUT Combinatorial)
T2 = 5 (INPUT Combinatorial)
T3 = 6 (INPUT Combinatorial)
T5 = 7 (INPUT Combinatorial)
T6 = 8 (INPUT Combinatorial)
T7 = 9 (INPUT Combinatorial)
T9 = 10 (INPUT Combinatorial)
CAS = 11 (INPUT Combinatorial)
TS = 13 (INPUT Combinatorial)
/LErr = 14 (INPUT Combinatorial)
/LMErr = 15 (INPUT Combinatorial)

/S_and_not_LEB = 23 (OUTPUT Active_Low Combinatorial)
/LEO_GenL = 22 (OUTPUT Active_Low Combinatorial)
LEI = 21 (OUTPUT Active_High Combinatorial)
/LEB = 20 (OUTPUT Active_Low Combinatorial)
CASI = 19 (OUTPUT Active_High Combinatorial)
/WE[0:2] = 16:18 (OUTPUT Active_Low Combinatorial);

DEFINE Read = /AC[0] * /AC[1] * AC[2],
Write = /AC[0] * AC[1] * /AC[2],
Long_Write = AC[0] * /AC[1] * /AC[2],
Refresh = AC[0] * /AC[1] * AC[2],
Scrub = /AC[0] * AC[1] * AC[2],
Init = AC[0] * AC[1] * AC[2];

BEGIN

ENABLE (LEB, LEI, LEO_GenL, S_and_not_LEB, WE[0:2], CASI);

LEB = Read * RASI * (/T2 * /LErr + /T3 * LErr);

LEI = Read * RASI * /T7 + Read * LErr * /LEO_GenL * /T9 +
Write * RASI * /T7 + Write * /LEO_GenL * /T9 + Long_Write * RASI * /T5 +
Scrub * RASI * /T7 + Scrub * /LEO_GenL * /T9;

LEO_GenL = (Scrub + Write + Read * LErr) * T3 * /T9 + Long_Write * RASI * /T2 + Init;

S_and_not_LEB = RASI * /T2 * (Read + Write + Long_Write + Refresh + Scrub) +
Read * RASI * (/T2 * /LErr + /T3 * LErr);

CASI = /T2 * CAS + TS * CAS;

WE[0] = Init + Long_Write * T5 * /T2 + (Scrub + Read * LErr + Write) * T6 * /T9 * /LMErr;
WE[1] = Init + Long_Write * T5 * /T2 + (Scrub + Read * LErr + Write) * T6 * /T9 * /LMErr;
WE[2] = Init + Long_Write * T5 * /T2 + (Scrub + Read * LErr + Write) * T6 * /T9 * /LMErr;

END.

```

"August 9, 1988

32-Bit Error Detection and Correction Board for the Micro Channel."

```

DEVICE                                INTERFACE (P22P10)                "U7"

PIN  /BE[0:3] = 1:4 (INPUT Combinatorial)
      T2 = 5 (INPUT Combinatorial)
      T10 = 6 (INPUT Combinatorial)
      /MemWr = 7 (INPUT Combinatorial)
      S_and_not_LEB = 8 (INPUT Combinatorial)
      AC[0:2] = 11:9 (INPUT Combinatorial)
      /Latched_Err = 13 (INPUT Combinatorial)
      LEDiag = 14 (INPUT Combinatorial)

      /LEY = 23 (OUTPUT Active_Low Combinatorial)
      /OE_SD[0:3] = 19:22 (OUTPUT Active_Low Combinatorial)
      /OE_EDC[0:3] = 18:15 (OUTPUT Active_Low Combinatorial);

DEFINE Read =      /AC[0] * /AC[1] * AC[2],
Write =      /AC[0] * AC[1] * /AC[2],
Scrub =      /AC[0] * AC[1] * AC[2],
Init =      AC[0] * AC[1] * AC[2];

BEGIN

ENABLE(OE_SD[0:3],OE_EDC[0:3],LG);

OE_EDC[0] = Init + Scrub * T2 * /T10 + Write * /BE[0] * T2 * /T10 + Read * Latched_Err
          * T2 * /T10 + OE_EDC[0] * /T10;

OE_EDC[1] = Init + Scrub * T2 * /T10 + Write * /BE[1] * T2 * /T10 + Read * Latched_Err
          * T2 * /T10 + OE_EDC[1] * /T10;

OE_EDC[2] = Init + Scrub * T2 * /T10 + Write * /BE[2] * T2 * /T10 + Read * Latched_Err
          * T2 * /T10 + OE_EDC[2] * /T10;

OE_EDC[3] = Init + Scrub * T2 * /T10 + Write * /BE[3] * T2 * /T10 + Read * Latched_Err
          * T2 * /T10 + OE_EDC[3] * /T10;

OE_SD[0] = LEDiag + S_and_not_LEB * (Write + Long_Write) * /OE_EDC[0];

OE_SD[1] = LEDiag + S_and_not_LEB * (Write + Long_Write) * /OE_EDC[1];

OE_SD[2] = S_and_not_LEB * (Write + Long_Write) * /OE_EDC[2];

OE_SD[3] = S_and_not_LEB * (Write + Long_Write) * /OE_EDC[3];

LEY = MemWr + LEDiag;

END.

```


"August 9, 1988

32-Bit Error Detection and Correction Board for the Micro Channel."

```

32-Bit Error Detection and Correction Board for the Micro Channel."
DEVICE          Output_Enable (P22P10)      "U8"
PIN             /BE[0:3] = 4:1 (INPUT Combinatorial)
               T7 = 5 (INPUT Combinatorial)
               /MemRd = 6 (INPUT Combinatorial)
               /MemWr = 7 (INPUT Combinatorial)
               /Err = 8 (INPUT Combinatorial)
               /SynSel = 9 (INPUT Combinatorial)
               /DiagSel = 10 (INPUT Combinatorial)
               /DMCSel = 11 (INPUT Combinatorial)
               /IOWr = 13 (INPUT Combinatorial)
               /IORD = 14 (INPUT Combinatorial)

               LEDiag = 23 (OUTPUT Active_High Combinatorial)
               /SynLE = 22 (OUTPUT Active_Low Combinatorial)
               /SynOE = 21 (OUTPUT Active_Low Combinatorial)
               RL = 20 (OUTPUT Active_Low Combinatorial)
               /OE_Bd[0:3] = 16:19 (OUTPUT Active_Low Combinatorial);

BEGIN

ENABLE (OE_Bd[0:3], SynLE, SynOE, RL, LEDiag);
ENABLE (IORD) = 0;

OE_Bd[0] = MemRd * T7 * BE[0];
OE_Bd[1] = MemRd * T7 * BE[1];
OE_Bd[2] = MemRd * T7 * BE[2];
OE_Bd[3] = MemRd * T7 * BE[3];
SynLE = (MemRd + MemWr) * Err;
SynOE = IORD * SynSel;
LEDiag = IOWr * DiagSel;
RL = DMCSel * IOWr;

END.

```

32-Bit Error Detection and Correction Board for the Micro Channel."

```

DEVICE          TIMER (P20RA10)      "U9"

PIN  /PRE_LOAD = 1 (CONTROL)
      Int1or2 = 2 (INPUT Combinatorial)
      Int3 = 3 (INPUT Combinatorial)
      Int5 = 4 (INPUT Combinatorial)
      Int6 = 5 (INPUT Combinatorial)
      Int7 = 6 (INPUT Combinatorial)
      Int8 = 7 (INPUT Combinatorial)
      Int9 = 8 (INPUT Combinatorial)
      Int10 = 9 (INPUT Combinatorial)
      /EoT = 10 (INPUT Combinatorial)
      /OE = 13 (CONTROL)

      T1 = 23 (OUTPUT Active_Low Registered)
      T2 = 22 (OUTPUT Active_Low Registered)
      T3 = 21 (OUTPUT Active_Low Registered)
      T5 = 20 (OUTPUT Active_Low Registered)
      T6 = 19 (OUTPUT Active_Low Registered)
      T7 = 18 (OUTPUT Active_Low Registered)
      T8 = 17 (OUTPUT Active_Low Registered)
      T9 = 16 (OUTPUT Active_Low Registered)
      T10 = 15 (OUTPUT Active_Low Registered);

BEGIN

ENABLE (T1, T2, T3, T6, T7, T8, T9, T10);

/T1 = 1;   CLOCK_PT(T1) = Int1or2;   PRESET(T1) = EoT;
/T2 = 1;   CLOCK_PT(T2) = Int1or2;   PRESET(T2) = EoT;
/T3 = 1;   CLOCK_PT(T3) = Int3;       PRESET(T3) = EoT;
/T5 = 1;   CLOCK_PT(T5) = Int5;       PRESET(T5) = EoT;
/T6 = 1;   CLOCK_PT(T6) = Int6;       PRESET(T6) = EoT;
/T7 = 1;   CLOCK_PT(T7) = Int7;       PRESET(T7) = EoT;
/T8 = 1;   CLOCK_PT(T8) = Int8;       PRESET(T8) = EoT;
/T9 = 1;   CLOCK_PT(T9) = Int9;       PRESET(T9) = EoT;
/T10 = 1;  CLOCK_PT(T10) = Int10;     PRESET(T10) = EoT;

END.

```

"August 9, 1988

32-Bit Error Detection and Correction Board for the Micro Channel."

DEVICE	Latch (16L8)	"U15"
PIN	/BE[3:0] = 1:4 (INPUT Combinatorial) Refresh = 5 (INPUT Combinatorial) ALE = 6 (INPUT Combinatorial) AC[2:0] = 7:10 (INPUT Combinatorial)	
	/BEI[3:0] = 19:16 (OUTPUT Active_Low Combinatorial) LongWord = 15 (OUTPUT Active_Low Combinatorial) /FRH = 14 (OUTPUT Active_Low Combinatorial) /FR = 13 (OUTPUT Active_Low Combinatorial);	
DEFINE	Refresh = AC[0] * /AC[1] * AC[2]; Scrub = /AC[0] * AC[1] * AC[2];	
BEGIN		
ENABLE	(BEI[0:3], LongWord, FRH, FR);	
BEI[0]	= BE[0] * /Cmd + BEI[0] * Cmd;	
BEI[1]	= BE[1] * /Cmd + BEI[1] * Cmd;	
BEI[2]	= BE[2] * /Cmd + BEI[2] * Cmd;	
BEI[3]	= BE[3] * /Cmd + BEI[3] * Cmd;	
/LongWord	= BE[3] * BE[2] * BE[1] * BE[0] + BEI[3] * BEI[2] * BEI[1] * BEI[0];	
FRH	= / (Refresh * /FRH + Refresh + Scrub);	
FR	= Refresh * FRH;	
END.		

DELAY LINE TAP CALCULATIONS

Derivation of the tap outputs is included here. The calculated time is adjusted to the nearest tap of the delay line (10-ns intervals) equal to or greater than the calculated time. The board is designed for 120-ns DRAMs.

	120 ns	100 ns	85 ns		120 ns	100 ns	85 ns
MSEL - RASI to MUX SELECT				INT4 - $\overline{\text{MERR}}$ from 29C660C and $\overline{\text{IOCHRDY}}$ for R/M/W			
t_{RAH} (DRAM) min	15.0	15.0	15.0	t_{PD} (RASI to $\overline{\text{RASn}}$) 29C668 max	27.0	27.0	27.0
t_{SKEW} ($\overline{\text{Qn}}$ to $\overline{\text{RASn}}$) 29C668 max	6.0	6.0	6.0	t_{ACC} DRAM max	120.0	100.0	85.0
Total	21.0	21.0	21.0	t_{PD} (Data In to $\overline{\text{MULT ERROR}}$) 29C660C max	20.0	20.0	20.0
CAS - RASI to CAS				t_{SU} ($\overline{\text{MERR}}$) 20RA10 min	13.0	13.0	13.0
MSEL	21.0	21.0	21.0	Total	180.0	160.0	145.0
t_{SKEW} ($\overline{\text{CASn}}$ to $\overline{\text{Qn}}$) 29C668 max	-2.0	-2.0	-2.0	INT 1 - $\overline{\text{IOCHRDY}}$ for Read without Error			
t_{ASC} DRAM min	0.0	0.0	0.0	t_{PD} (RASI to $\overline{\text{RASn}}$) 29C668 max	27.0	27.0	27.0
$-t_{\text{PD}}$ ($\overline{\text{CAS}}$ to $\overline{\text{CASI}}$) 22P8B min	-6.0	-6.0	-6.0	t_{ACC} DRAM max	120.0	100.0	85.0
Total	13.0	13.0	13.0	t_{PD} (Data In to Data Out) 29C660C max	24.0	24.0	24.0
INT5 - Valid Check Bits on Long Write				t_{PD} (Data Out to System Data) 29C983 max	14.0	14.0	14.0
t_{PD} (RASI to LEO_GenL) 22P10 max	15.0	15.0	15.0	$-t_{\text{PD}}$ (T1 to $\overline{\text{IOCHRDY}}$) 20L10B min	-6.0	-6.0	-6.0
t_{PD} (LEO_GenL to SC) 29C660C max	18.0	18.0	18.0	$-t_{\text{PD}}$ (INT1 to T1) 20RA10 min	-7.0	-7.0	-7.0
$-t_{\text{PD}}$ (T5 to WE) 22P10 min	-7.0	-7.0	-7.0	Total	172.0	152.0	137.0
Total	26.0	26.0	26.0	INT 6 - Corrected Data and Check Bits (R/M/W)			
INT7 - Data Valid to 29C660C				INT3	164.0	144.0	129.0
t_{PD} (RASI to $\overline{\text{RASn}}$) 29C668 max	27.0	27.0	27.0	t_{SKEW} (T6 to T3) 20RA10 max	0.5	0.5	0.5
t_{ACC} DRAM max	120.0	100.0	85.0	t_{PD} (T3 to LEO_GEN) 20L8B max	15.0	15.0	15.0
$-t_{\text{PD}}$ (INT7 to T7) 20RA10 min	-7.0	-7.0	-7.0	t_{PD} (LEO_GEN to SCn) 29C660C max	18.0	18.0	18.0
Total	140.0	120.0	105.0	t_{DS} DRAM min	0.0	0.0	0.0
INT2 - $\overline{\text{ERROR}}$ from 29C660C				Total	197.5	177.5	162.5
t_{PD} (RASI to $\overline{\text{RASn}}$) 29C668 max	27.0	27.0	27.0	TS - Pulsed $\overline{\text{CAS}}$			
t_{ACC} DRAM max	120.0	100.0	85.0	INT6	197.5	177.5	162.5
t_{PD} (Data In to $\overline{\text{ERROR}}$) 29C660C max	16.0	16.0	16.0	t_{PD} (INT6 to T6) 20RA10 max	20.0	20.0	20.0
t_{SU} ($\overline{\text{ERROR}}$) 20RA10 min	13.0	13.0	13.0	t_{SKEW} ($\overline{\text{WEn}}$ to $\overline{\text{CASI}}$) 20L8B	3.0	3.0	3.0
Total	176.0	156.0	141.0	$-t_{\text{PD}}$ ($\overline{\text{CASI}}$ to $\overline{\text{CASn}}$) 29C668 min	-15.0	-15.0	-15.0
INT3 - Corrected Data from EDC				t_{WCS} DRAM	0.0	0.0	0.0
t_{PD} (RASI to $\overline{\text{RASn}}$) 29C668 max	27.0	27.0	27.0	Total	205.5	185.5	170.5
t_{ACC} DRAM max	120.0	100.0	85.0	INT9 - End of $\overline{\text{WEn}}$ and RASI (R/M/W)			
t_{PD} (Data In-Data Out) 29C660C max	24.0	24.0	24.0	t_{S}	205.5	185.5	170.5
$-t_{\text{PD}}$ (INT3 to T3) 20RA10 min	-7.0	-7.0	-7.0	t_{PD} (TS to $\overline{\text{CASI}}$) 20L8B max	15.0	15.0	15.0
Total	164.0	144.0	129.0	t_{PD} ($\overline{\text{CASI}}$ to $\overline{\text{CASn}}$) 29C668 max	31.0	31.0	31.0
				t_{WCH} ($\overline{\text{WE}}$ Pulse Width) DRAM min	25.0	25.0	25.0
				$-t_{\text{PD}}$ (T9 to WE) 20L8B min	-7.0	-7.0	-7.0
				$-t_{\text{PD}}$ (INT9 to T9) 20RA10 min	-7.0	-7.0	-7.0
				Total	262.5	242.5	227.5

120 ns 100 ns 85 ns

PARTS LIST (CONT.)

INT8 - End of Read without Error

INT2	176.0	156.0	141.0
t_{RP} RAM min	90.0	80.0	70.0
Total	266.0	236.0	211.0

INT10 - End of R/M/W Cycle

INT9	262.5	242.5	227.5
t_{RP} RAM min	90.0	80.0	70.0
Total	352.5	322.5	297.5

Signal	Should Be (ns)	Is (ns)	Note
MSEL	21.0	30.0	
CAS	22.0	30.0	= MSEL - 8
INT5	26.0	30.0	
INT7	140.0	140.0	
INT2	176.0	180.0	
INT3	164.0	170.0	
INT4	180.0	180.0	
INT1	172.0	180.0	
INT6	203.5	210.0	= INT3 + 33.5
TS	215.5	220.0	= INT6 + 8
INT8	270.0	270.0	= INT2 + 90
INT9	278.0	280.0	= TS + 57
INT10	370.0	370.0	= INT9 + 90

Notes:

1. Table for 120 ns DRAMs
2. Tap which are dependent or related to other taps are indicated with a comment in the "explanation" column.
3. Timing figures are based on Am29C660C data.

PARTS LIST

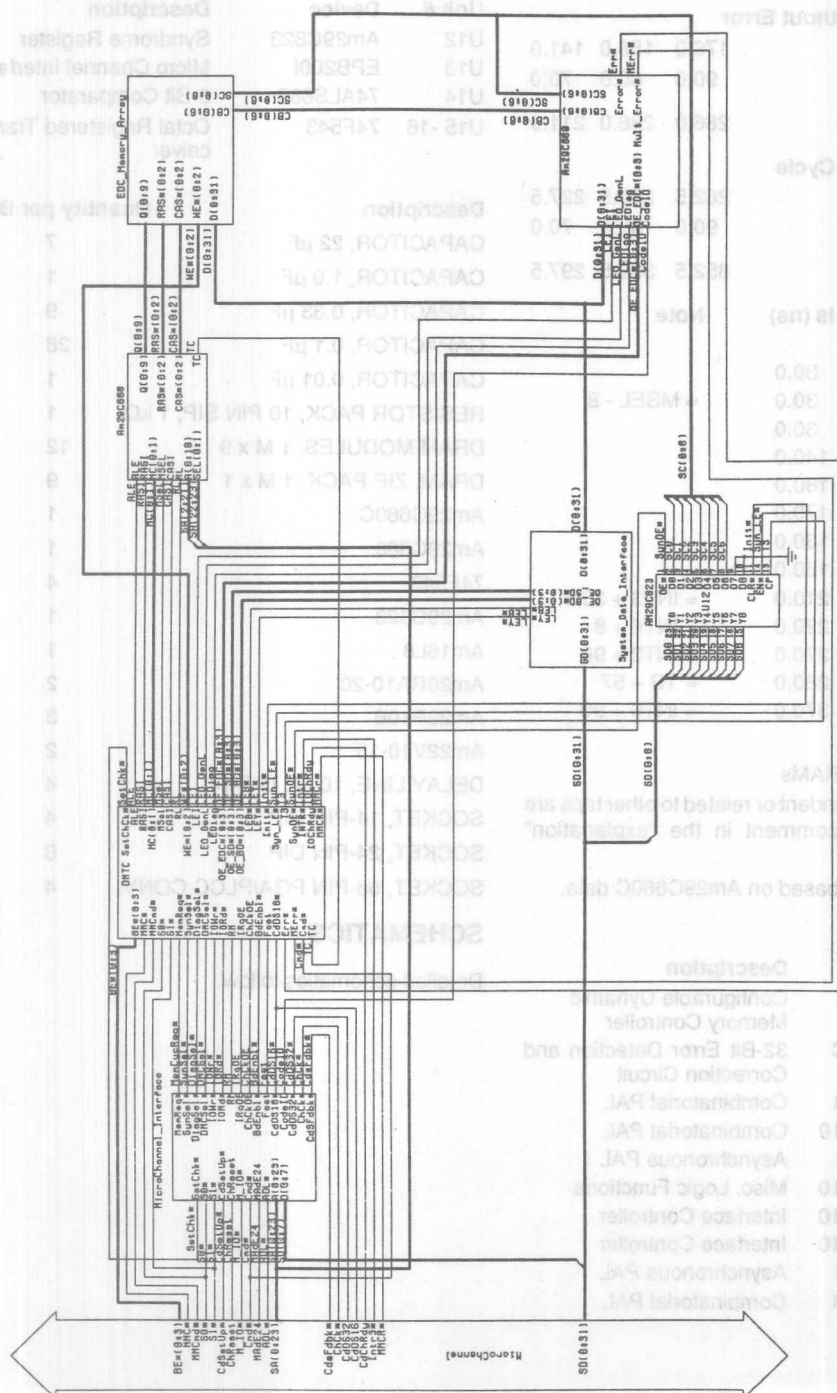
Unit #	Device	Description
U1	Am29C668	Configurable Dynamic Memory Controller
U2	Am29C660C	32-Bit Error Detection and Correction Circuit
U3	AmPAL16L8	Combinatorial PAL
U4	AmPAL22V10	Combinatorial PAL
U5	PAL20RA10	Asynchronous PAL
U6	AmPAL22P10	Misc. Logic Functions
U7	AmPAL22P10	Interface Controller
U8	AmPAL22P10	Interface Controller
U9	PAL20RA10	Asynchronous PAL
U10	AmPAL16L8	Combinatorial PAL

Unit #	Device	Description
U12	Am29C823	Syndrome Register
U13	EPB200I	Micro Channel Interface
U14	74ALS688	8-Bit Comparator
U15 -18	74F543	Octal Registered Transceiver

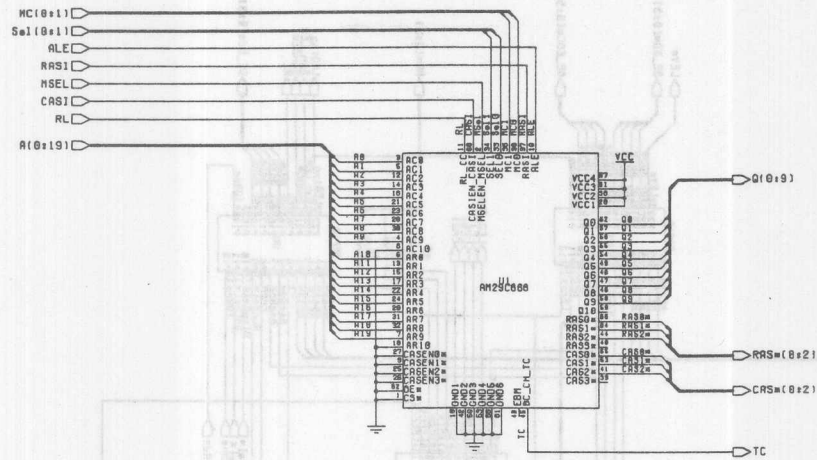
Description	Quantity per Board
CAPACITOR, 22 μ F	7
CAPACITOR, 1.0 μ F	1
CAPACITOR, 0.33 μ F	9
CAPACITOR, 0.1 μ F	28
CAPACITOR, 0.01 μ F	1
RESISTOR PACK, 10 PIN SIP, 1 k Ω	1
DRAM MODULES, 1 M x 9	12
DRAM, ZIP PACK, 1 M x 1	9
Am29C660C	1
Am29C668	1
74F543	4
Am29C823	1
Am16L8	1
Am20RA10-20	2
Am22P10B	3
Am22V10-15	2
DELAY LINE, 10 ns, DIP-14	4
SOCKET, 14-PIN DIP	4
SOCKET, 24-PIN DIP	8
SOCKET, 68-PIN PGA/PLCC CONV.	4

SCHEMATICS

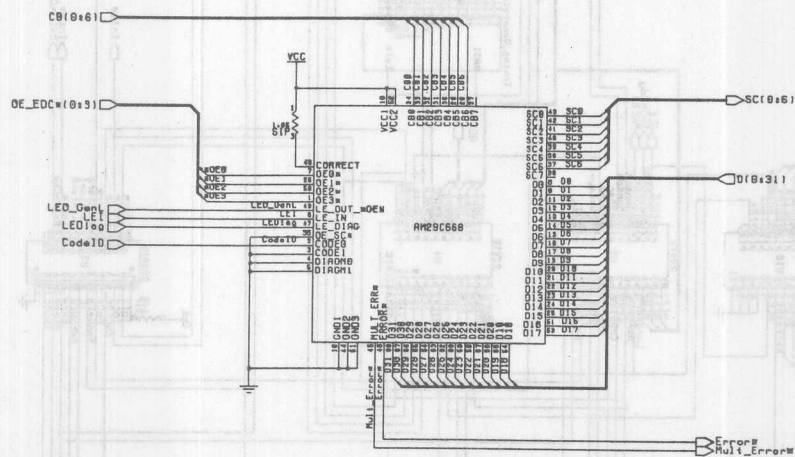
Detailed schematics follow.



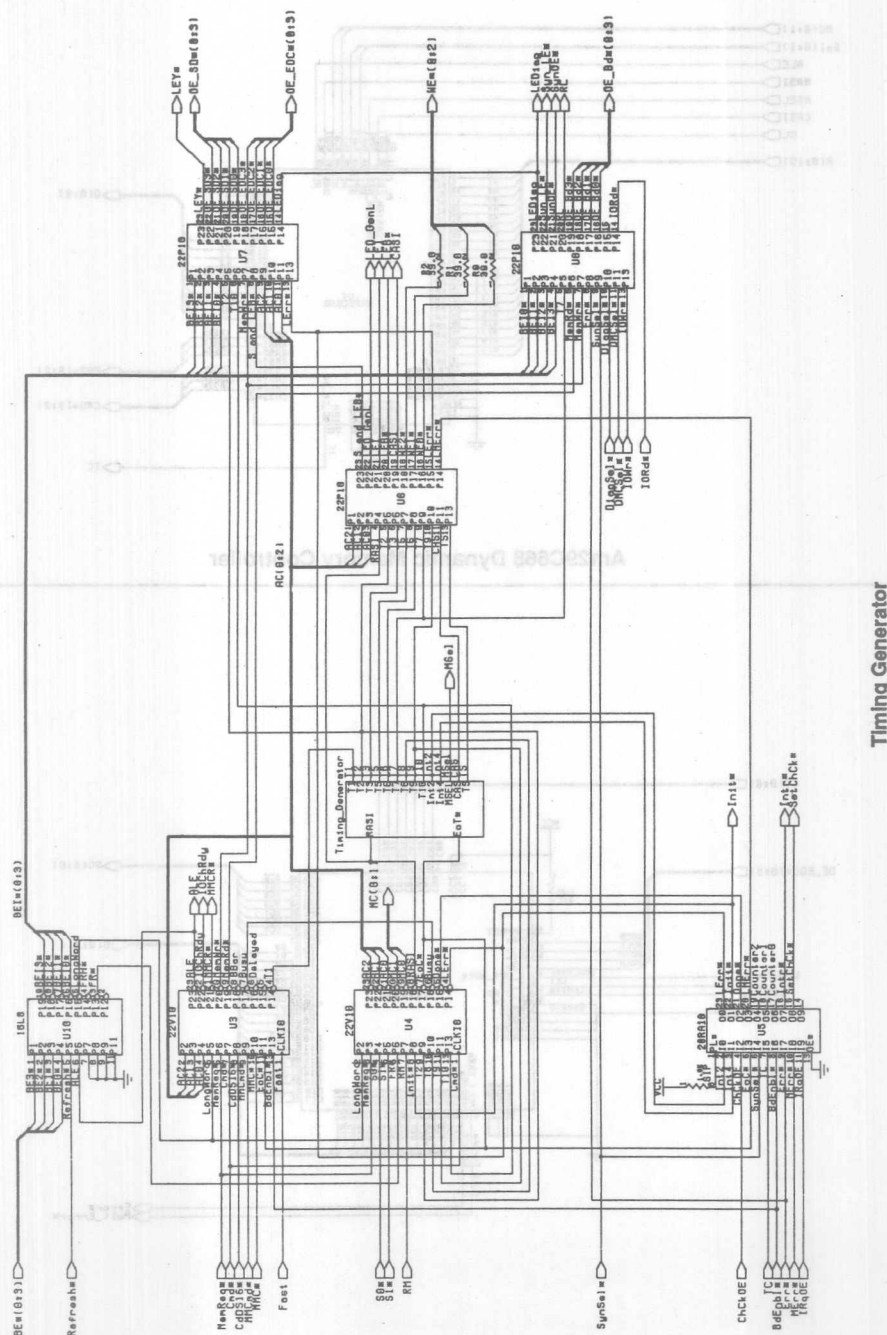
Top Level Schematic

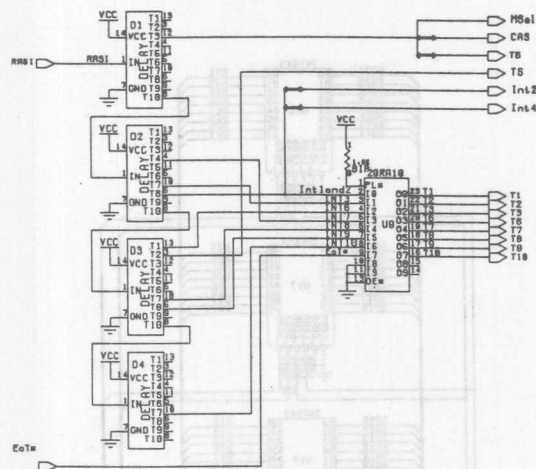


Am29C668 Dynamic Memory Controller

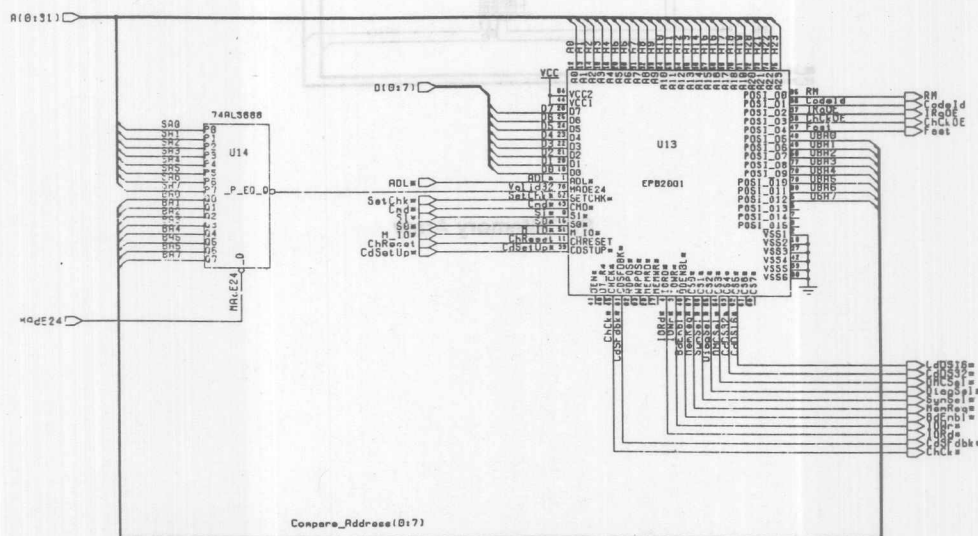


Error Detection and Correction

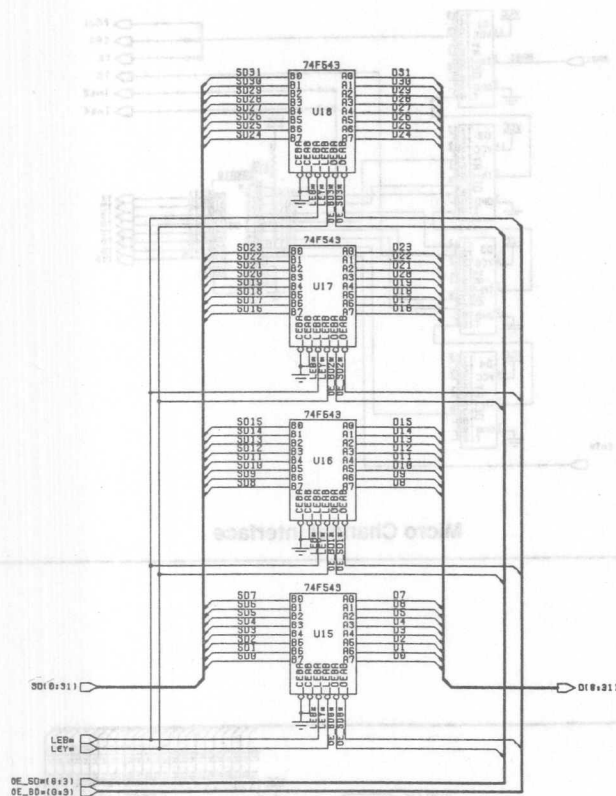




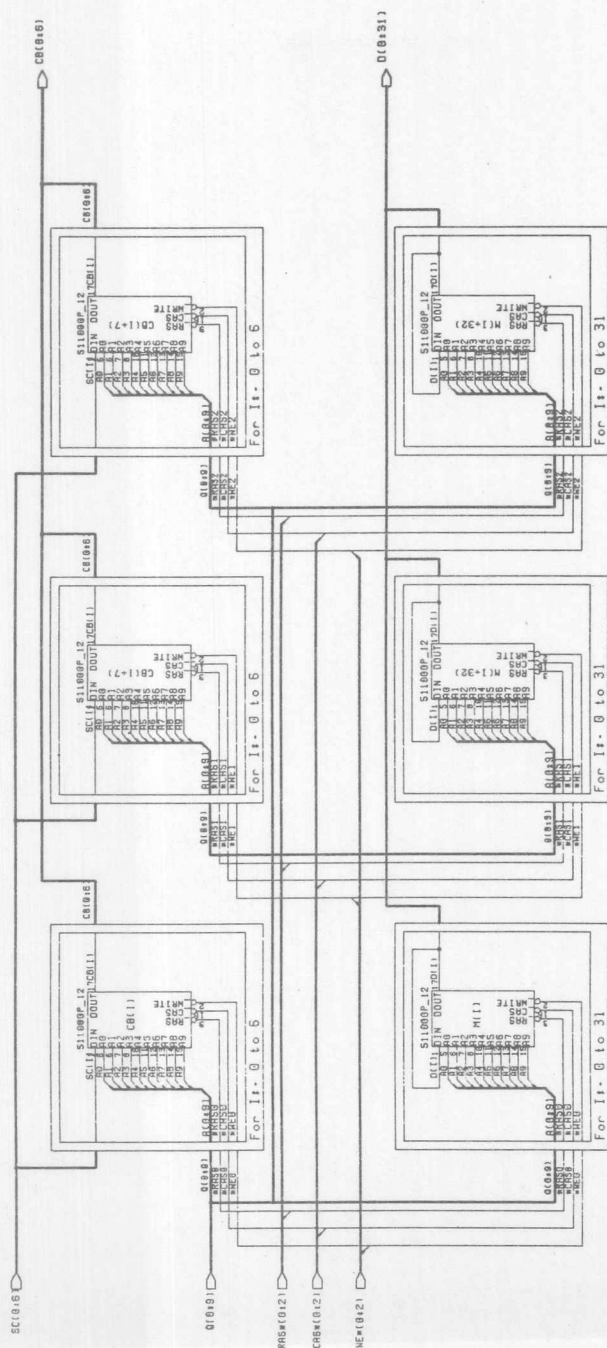
Micro Channel Interface



System Data Interface



EDC Memory Array



Dynamic Memory Timing Controller



CHAPTER 5

Special Applications and Article Reprints

Introduction	5-2
Configurable DRAM Controller Enhances System Performance	5-3
Four-Megabit DRAM Controller Offers Burst Addressing	5-11
High-Speed VCEP Demonstration Board Using the Am29C668 CDMC	5-15
Universal Coprocessor Platform (UCP) and Network Terminator-S Interface (NTs)	5-19



Special Applications and Article Reprints

INTRODUCTION

This chapter contains three article reprints, two of which were originally printed in European publications and translated for use in this data book. These two articles discuss the Am29C668 Configurable Dynamic Memory Controller (CDMC) in detail. The third article is a reprint from selected section of the Universal Coprocessor Platform (UCP) and Network Terminator-S Interface (NTs) technical manual which is applicable to the Am29C668 CDMC and the Am29C983 MBE.

A special application article is also included describing a demonstration board using the Am95C71 Video-Data Compression/Expansion Processor (VCEP) and the Am29C668 CDMC. The board requires a dedicated memory buffer to hold compressed images, which is designed using DRAMs controlled by the Am29C668 CDMC.

Configurable DRAM Controller Enhances System Performance

by Percy R. Aria, Senior Product Planner

INTRODUCTION

With today's evolution of fast processors and RISC architectures, memory speed is a major factor in system throughput. Very fast memory in the form of static RAM looks attractive at first glance, but as the memory size increases, it becomes far less attractive due to prohibitive cost per bit compared to dynamic RAM. DRAM requires more complex control compared to SRAM, but cost per bit and high densities make DRAM very attractive for a wide range of applications.

The Am29C668 4-Mbit Configurable Dynamic Memory Controller greatly simplifies DRAM design. Because the Am29C668 is highly integrated as well as configurable, it can be used with virtually any processor in any system architecture. A block diagram of the Am29C668 is shown in Figure 1.

OPERATING MODES

The Am29C668 has two basic modes of operation, Read/Write and refresh; the timing diagram is shown in Figure 2. In the Read/Write mode, the Am29C668 latches the column, row and bank addresses. It then multiplexes the row and column addresses to the DRAMs under the

control of either internally generated timing signals (auto-timing mode) or externally generated input signals.

The row address is latched in the DRAMs by the active (Low-going) edge of the Row Address Strobe $\overline{\text{RAS}}$ output, which follows the active (High-going) edge of the Row Address Strobe Input RASI . The address lines are then switched to the column address by either an internally generated signal if auto timing is selected or by pulling the Multiplexer Select MSEL signal active High if external timing is selected.

Read/Write Mode Optimization

The Read/Write mode of the Am29C668 may be optimized for the shortest DRAM access time in three different ways, depending on the system environment, software requirements and hardware configuration. These are Burst/Block-Mode Access, Cache-Mode Access and Bank-Interleave Mode Access.

Burst/Block-Mode Access

In this access mode, the Am29C668 can operate with processors that request burst accesses. Burst/block transfer is used by a high-performance processor to fill

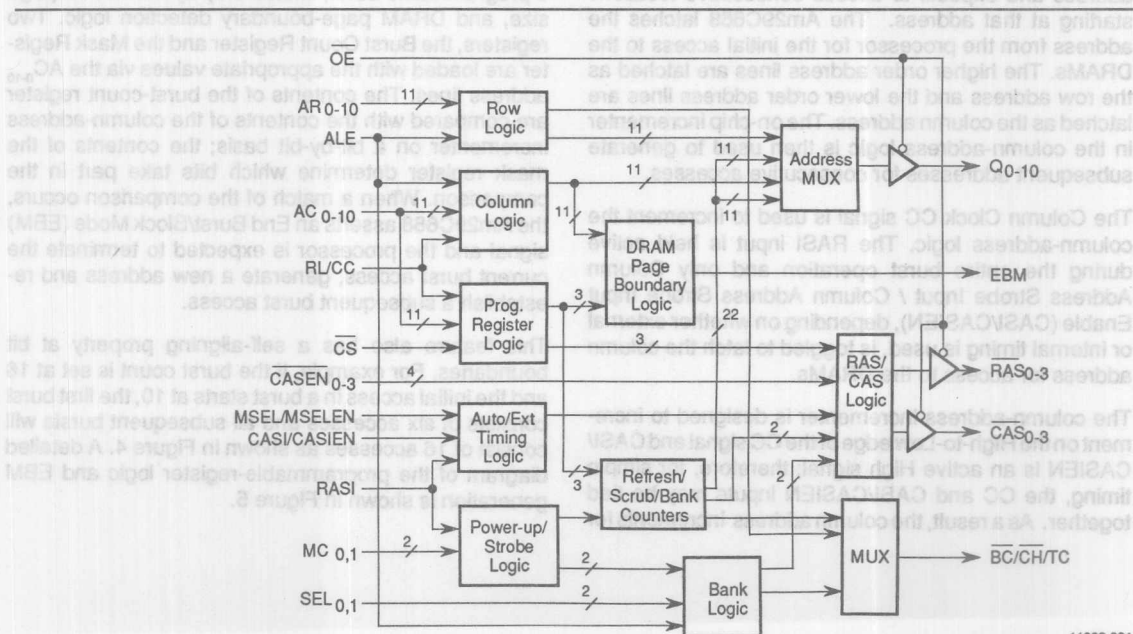


Figure 1. Am29C668 Block Diagram

11902-001A

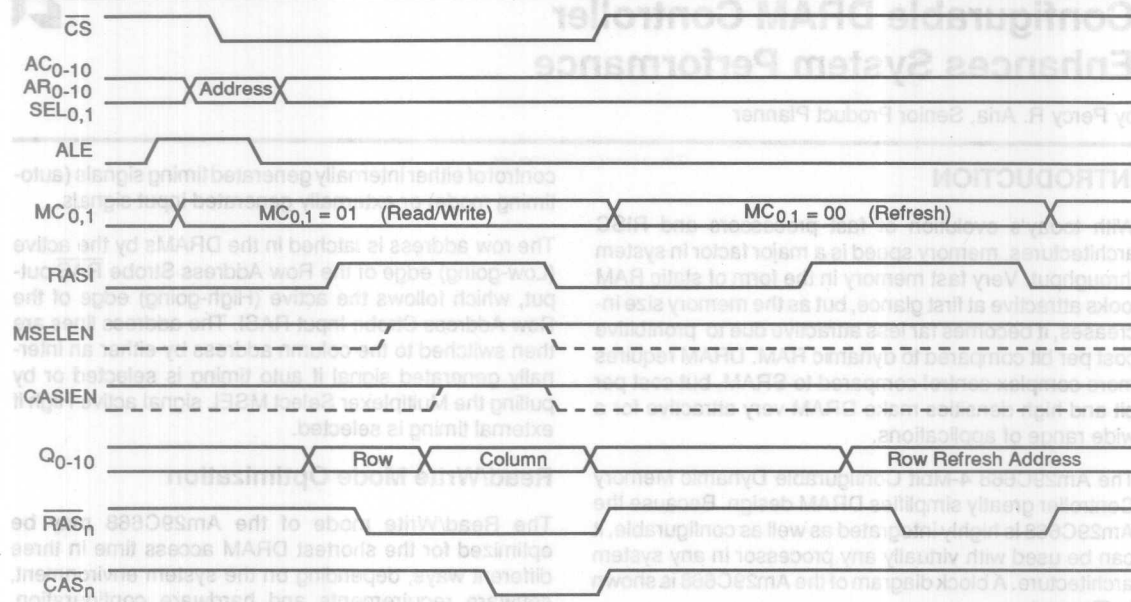


Figure 2. Read/Write and Refresh Operations

11902-002A

transfer is used by a high-performance processor to fill the cache, when a cache miss is detected; it is also used to support data pipelining.

During a burst/block access, the processor generates an address and expects to access consecutive locations starting at that address. The Am29C668 latches the address from the processor for the initial access to the DRAMs. The higher order address lines are latched as the row address and the lower order address lines are latched as the column address. The on-chip incrementer in the column-address logic is then used to generate subsequent addresses for consecutive accesses.

The Column Clock CC signal is used to increment the column-address logic. The RASi input is held active during the entire burst operation and only Column Address Strobe Input / Column Address Strobe Input Enable (CASI/CASIEN), depending on whether external or internal timing is used, is toggled to latch the column address for access to the DRAMs.

The column-address incrementer is designed to increment on the High-to-Low edge of the CC signal and CASI/CASIEN is an active High signal; therefore, for simple timing, the CC and CASI/CASIEN inputs may be tied together. As a result, the column address increments for

the next access at the end of the DRAM access, when the CASI/CASIEN signal is deactivated. Figure 3 shows the timing for this type of access.

Additional support for the burst/block access consists of a programmable burst, limited only by the DRAM page size, and DRAM page-boundary detection logic. Two registers, the Burst Count Register and the Mask Register are loaded with the appropriate values via the AC₀₋₁₀ address lines. The contents of the burst-count register are compared with the contents of the column-address incrementer on a bit-by-bit basis; the contents of the mask register determine which bits take part in the comparison. When a match of the comparison occurs, the Am29C668 asserts an End Burst/Block Mode (EBM) signal and the processor is expected to terminate the current burst access, generate a new address and re-establish a subsequent burst access.

This feature also has a self-aligning property at bit boundaries. For example, if the burst count is set at 16 and the initial access in a burst starts at 10, the first burst consists of six accesses and all subsequent bursts will consist of 16 accesses as shown in Figure 4. A detailed diagram of the programmable-register logic and EBM generation is shown in Figure 5.

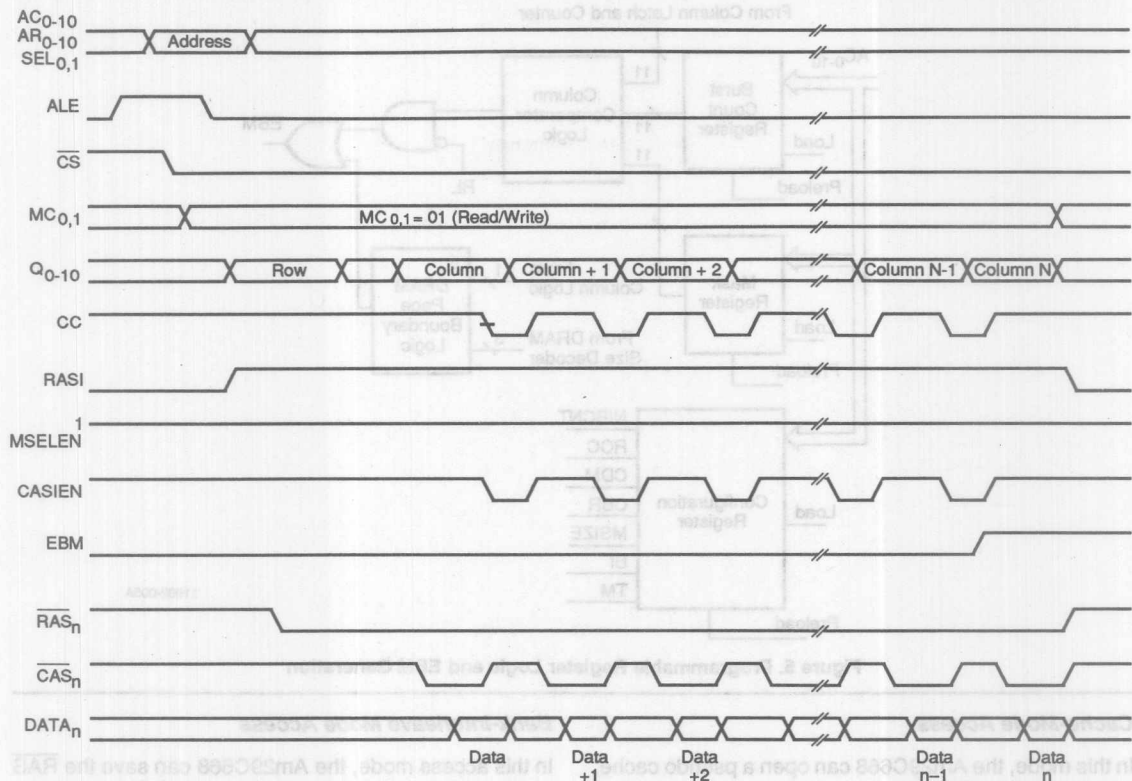


Figure 3. Burst Mode Access Ended by the Am29C668 (Auto Timing with External Override)

11902-003A

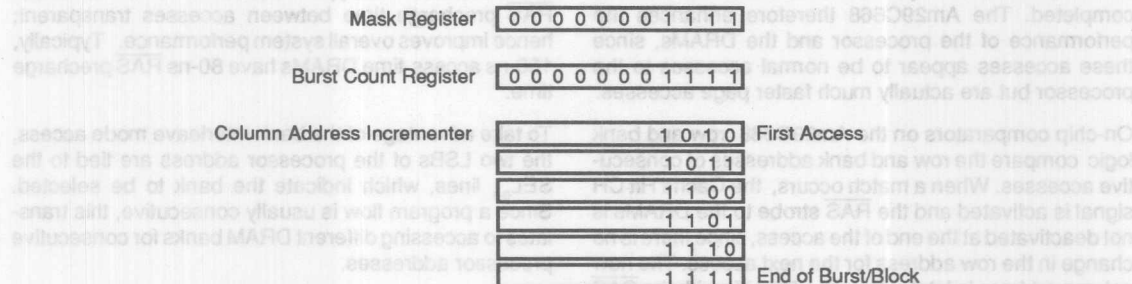


Figure 4. Burst-Mode Self-Alignment Example

11902-004A

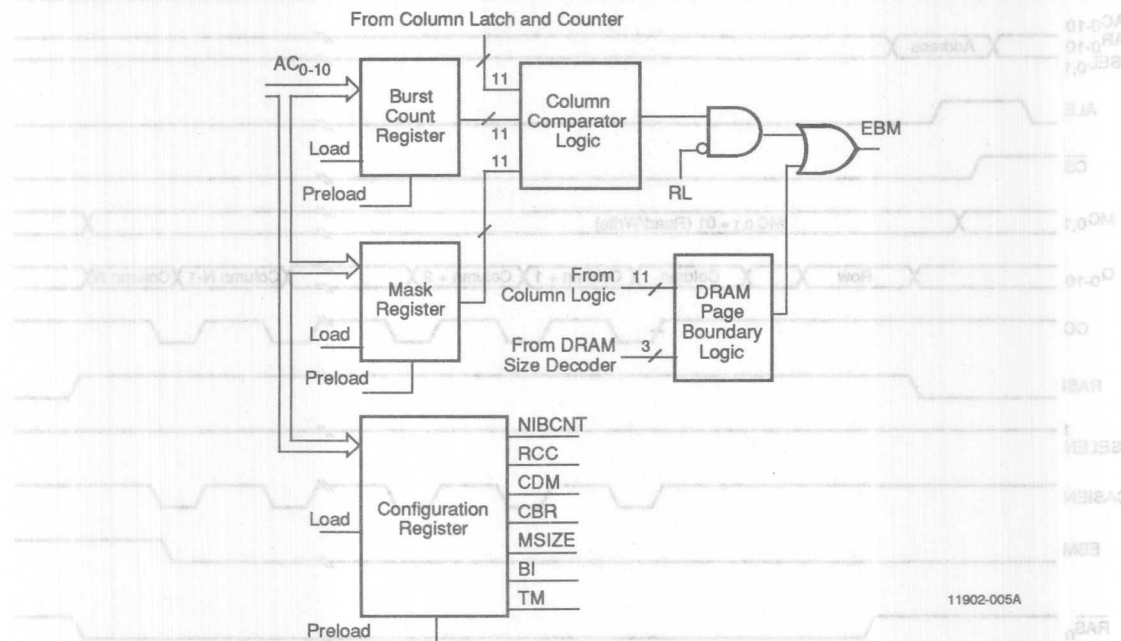


Figure 5. Programmable Register Logic and EBM Generation

Cache-Mode Access

In this mode, the Am29C668 can open a pseudo cache, the size of the DRAM page, and make fast random accesses to locations within the page. This access mode eliminates the $\overline{\text{RAS}}$ precharge time and the $\overline{\text{RAS}}$ access time from the DRAM cycle time, once the first access is completed. The Am29C668 therefore enhances the performance of the processor and the DRAMs, since these accesses appear to be normal accesses to the processor but are actually much faster page accesses.

On-chip comparators on the Am29C668 row and bank logic compare the row and bank addresses of consecutive accesses. When a match occurs, the Cache Hit CH signal is activated and the $\overline{\text{RAS}}$ strobe to the DRAMs is not deactivated at the end of the access, since there is no change in the row address for the next access. The new column address is latched into the DRAMs with the $\overline{\text{CAS}}$ strobe. Figure 6 shows a timing diagram for this type of access. The data from the DRAM is then available after the $\overline{\text{CAS}}$ access time of the DRAMs. The CH signal remains active as long as there is a match on the comparison of the row and bank addresses of subsequent accesses. When a mismatch occurs, the CH signal is deactivated and the external timing generator deactivates the $\overline{\text{RAS}}$ signal, which in turn latches the new row and bank addresses for future comparison. The $\overline{\text{RAS}}$ signal to the DRAMs is also deactivated and goes through precharge before starting the new access.

Bank-Interleave Mode Access

In this access mode, the Am29C668 can save the $\overline{\text{RAS}}$ precharge time between consecutive accesses to different DRAM banks, by overlapping the $\overline{\text{RAS}}$ precharge time of the previous access with the access time of the current access. This type of access therefore makes the $\overline{\text{RAS}}$ precharge time between accesses transparent; hence improves overall system performance. Typically, 100-ns access-time DRAMs have 80-ns $\overline{\text{RAS}}$ precharge time.

To take advantage of the bank-interleave mode access, the two LSBs of the processor address are tied to the $\text{SEL}_{0,1}$ lines, which indicate the bank to be selected. Since a program flow is usually consecutive, this translates to accessing different DRAM banks for consecutive processor addresses.

In this mode, the Am29C668 makes a comparison of the consecutive DRAM bank addresses. When a mismatch occurs, the Bank Interleave BI signal goes active and the $\overline{\text{RAS}}$ strobe, generated by an external timing generator, to the new bank is generated right away, while the $\overline{\text{RAS}}$ strobe to the previous bank is going through a precharge. If, however, a match occurs, the BI signal goes inactive. This means that the current access is to the same bank as the previous access, in which case the $\overline{\text{RAS}}$ strobe is deactivated and goes through precharge before being activated for the current access.

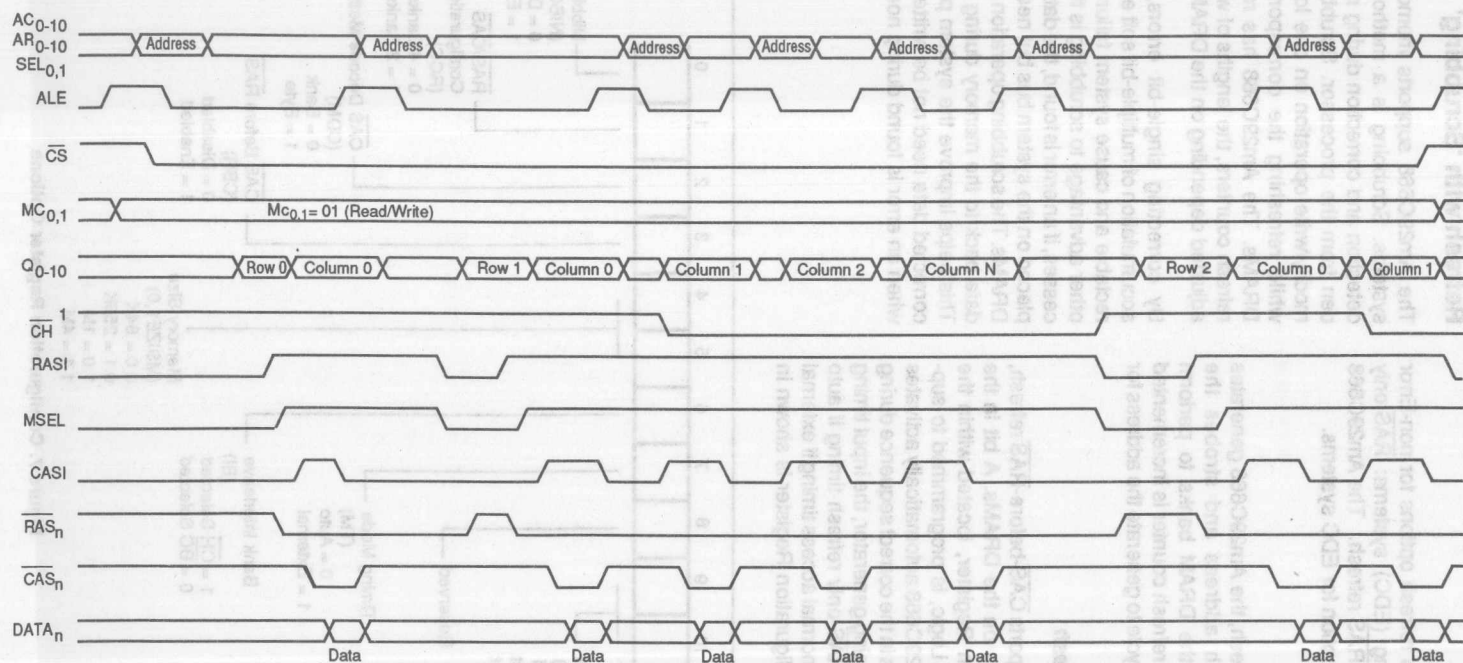


Figure 6. Cache-Mode Access with Page Mode DRAMs (External Timing)

Refresh Options

The Am29C668 has two refresh options for non-Error Detecting and Correcting (EDC) systems: $\overline{\text{RAS}}$ -only refresh and $\overline{\text{CAS}}$ -before- $\overline{\text{RAS}}$ refresh. The Am29C668 also offers a scrubbing option for EDC systems.

$\overline{\text{RAS}}$ -Only Refresh

During the $\overline{\text{RAS}}$ -only refresh, the Am29C668 generates the appropriate refresh address and strobes the corresponding $\overline{\text{RAS}}$ to the DRAM banks to perform refresh. The Am29C668 refresh counter is incremented at the end of the refresh cycle to generate the address for the next refresh cycle.

$\overline{\text{CAS}}$ -Before- $\overline{\text{RAS}}$ Refresh

The Am29C668 also supports $\overline{\text{CAS}}$ -before- $\overline{\text{RAS}}$ refresh, if this feature is available on the DRAMs. A bit in the Am29C668 Configuration Register, located within the Programmable Register Logic, is programmed to support this feature. The Am29C668 automatically activates the $\overline{\text{CAS}}$ and $\overline{\text{RAS}}$ strobes in the correct sequence during refresh. To the system timing generator, the input timing looks the same as the $\overline{\text{RAS}}$ -only refresh timing if auto timing is used, or like the normal access timing if external timing is used. The Configuration Register is shown in Figure 7.

Refresh with 'Scrubbing'

The Am29C668 supports memory scrubbing in EDC systems. Scrubbing is a method of performing error detection and correction during refresh operations hidden from the processor. Scrubbing performs a read/modify/write operation on one location in the memory while refreshing the corresponding row on all the DRAMs. The Am29C668 has row, column and bank refresh counters, the lengths of which are automatically adjusted depending on the DRAM size being used.

By correcting single-bit errors, scrubbing prevents accumulation of multiple-bit soft errors, which are uncorrectable and cause system failure and down time. Another advantage to scrubbing is that during normal accesses, if an error is found, the data can be corrected and placed on the system bus but need not be written to the DRAMs. The scrubbing operation can write the corrected data back to the memory during the refresh operation. This helps improve the system performance, since the corrected data need not be written back to the memory when an error is found during normal accesses.

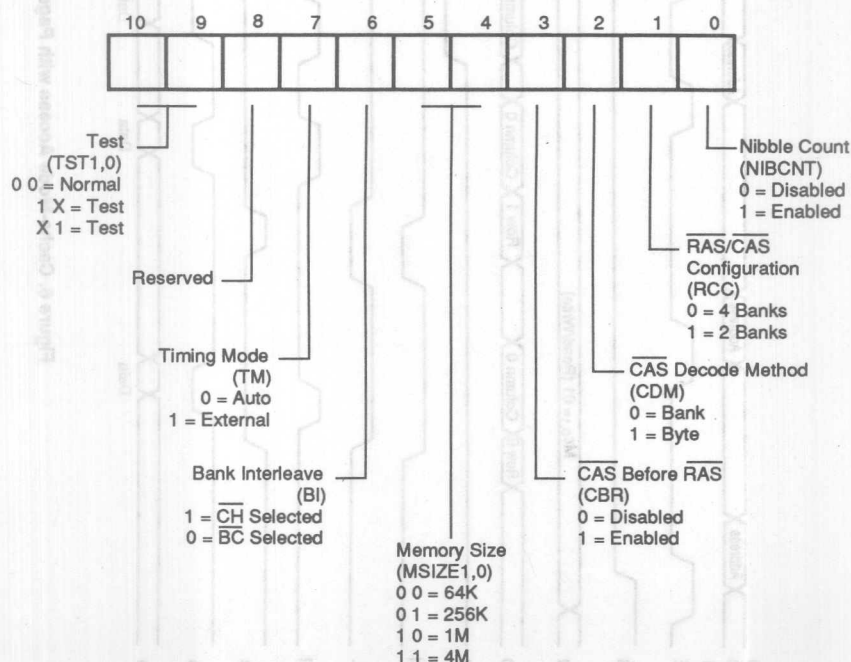


Figure 7. Configuration Register Options

11902-007A

Timing Options

Auto-Timing

In the auto-timing mode, the Am29C668 can be configured to generate its own internal MSEL and CASI timing signals for the output DRAM strobes (RAS_n and CAS_n) from the input RASI signal. The auto-timing is optimized for 100-ns DRAMs.

External Timing

An external timing mode is also available so that the user can externally generate the MSEL and CASI timing signals for a specific application or DRAM.

Auto-Timing With External Override

A third option consists of auto-timing with an external override. In this mode, the MSEL and CASI inputs are defined as MSEL Enable MSELEN and CASI Enable CASIEN, respectively, and are ANDed with their respectively generated internal signals. This option can be used effectively for nibble-mode DRAMs, when the initial access can be made using auto-timing. The RASI signal is kept active and the CASIEN signal can simply be

toggled to make the remaining three accesses of the nibble. A timing diagram of this type of an access is shown in Figure 8. A similar method can be used in the Burst/Block Access Mode and the Cache Access Mode.

OTHER DISTINCTIVE FEATURES

Nibble-Access Support

The incrementer on the Am29C668 can be configured to perform a modulo-four (nibble) count. A modulo-four burst access may be performed with a single address from the processor. Figure 9 shows a timing diagram for the nibble-mode access. This makes ordinary page-mode DRAMs look like nibble-mode DRAMs to the processor. Auto timing with external override is particularly well suited for this type of operation.

Configurable Drive Capability

The Am29C668 can be configured to drive two or four banks of DRAMs by programming a bit in the Configuration Register. By configuring it for two banks the drive capability of the RAS and CAS strobes is doubled.

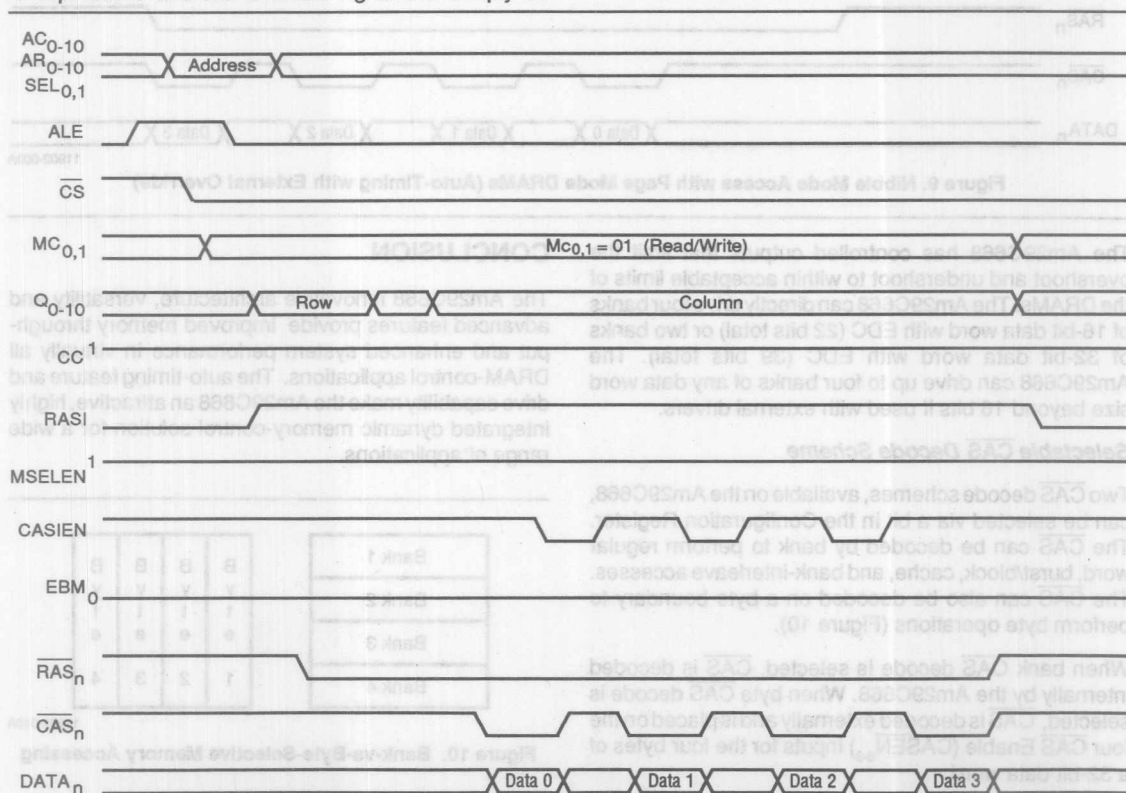


Figure 8. Nibble Mode Access with Nibble Mode DRAMs (Auto-Timing with External Override)

11902-008A

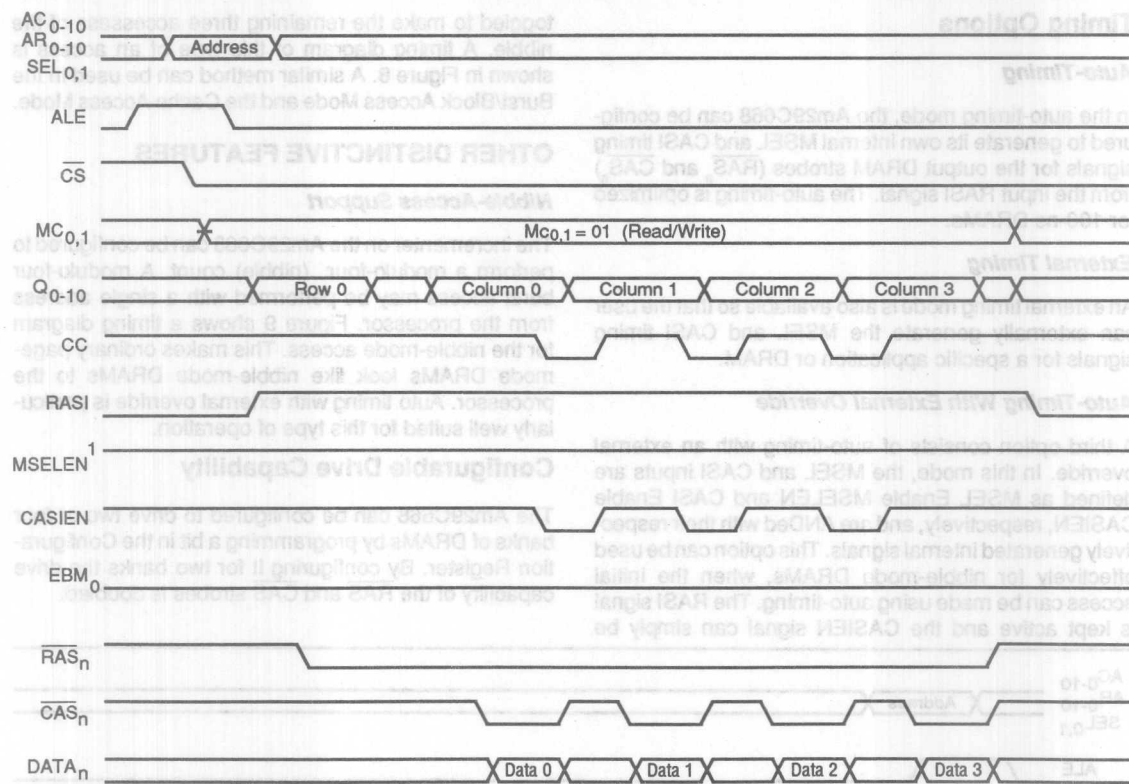


Figure 9. Nibble Mode Access with Page Mode DRAMs (Auto-Timing with External Override)

The Am29C668 has controlled outputs that limit the overshoot and undershoot to within acceptable limits of the DRAMs. The Am29C668 can directly drive four banks of 16-bit data word with EDC (22 bits total) or two banks of 32-bit data word with EDC (39 bits total). The Am29C668 can drive up to four banks of any data word size beyond 16 bits if used with external drivers.

Selectable $\overline{\text{CAS}}$ Decode Scheme

Two $\overline{\text{CAS}}$ decode schemes, available on the Am29C668, can be selected via a bit in the Configuration Register. The $\overline{\text{CAS}}$ can be decoded by bank to perform regular word, burst/block, cache, and bank-interleave accesses. The $\overline{\text{CAS}}$ can also be decoded on a byte boundary to perform byte operations (Figure 10).

When bank $\overline{\text{CAS}}$ decode is selected, $\overline{\text{CAS}}$ is decoded internally by the Am29C668. When byte $\overline{\text{CAS}}$ decode is selected, $\overline{\text{CAS}}$ is decoded externally and is placed on the four $\overline{\text{CAS}}$ Enable ($\text{CAS1EN}_{0,3}$) inputs for the four bytes of a 32-bit data word.

CONCLUSION

The Am29C668 innovative architecture, versatility and advanced features provide improved memory throughput and enhanced system performance in virtually all DRAM-control applications. The auto-timing feature and drive capability make the Am29C668 an attractive, highly integrated dynamic memory-control solution for a wide range of applications.

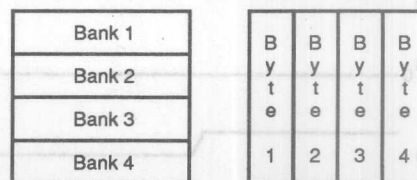


Figure 10. Bank-vs-Byte-Selective Memory Accessing



Four-Megabit DRAM Controller Offers Burst Addressing

by Bo Molander, Senior Field Applications Engineer

The advantages of dynamic random-access memory (DRAM) are lowest cost per bit compared to other semiconductor memories, and small packaging that requires minimum board space. Conversely, the best known drawback of DRAMs is that they must be refreshed every second or fourth millisecond to preserve the data stored in the array. In addition, to take advantage of the small package, memory row and column addresses must be multiplexed onto the input pins. Over the years, a number of different DRAM controllers have been introduced that make these drawbacks transparent to the DRAM user.

Initially, simple building blocks were offered, each device performing only a part of the total DRAM-control function. In time, however, more integrated solutions were presented. Soon, two major pathways were chosen: in the first, no timing-generation support was included in the memory controller; in the second, timing delays for the RAS and CAS strobes were generated within the controller. Both DRAM-controller types found applications dictated by system requirements. In the past, DRAM controllers with no internal timing generation were used extensively in systems with very fast memories requiring the shortest possible access times. A 1-Mbit DRAM controller, such as the Am29368, can be tailored to a specific memory system according to the number of memory devices, associated capacitance and the resulting propagation delays. For example, the Am29368 DRAM controller can be used very efficiently with 85-ns DRAMs.

A DRAM controller with internal timing generation provides a more compact solution with fewer devices. Unfortunately, speed and flexibility are sometimes sacrificed. These older DRAM controllers are simply too slow to keep up with today's dynamic memories that have access times from 60 to 100 ns.

The limitations of these early memory controllers have driven the system designer to solving memory-control requirements with PAL® devices or to other similar programmable devices. A simple memory-control function is easily implemented with a couple of PAL devices; but, more sophisticated features, like block transfers and burst addressing, increase the required number of PAL devices. The final memory controller could simply occupy too much board space.

The Am29C668 Configurable Dynamic Memory Controller/Driver (CDMC), shown in Figure 1, gives the memory-system designer a new alternative. It is fabricated using CMOS technology to provide very high speed and low power consumption, and can directly control DRAMs up to 4 Mbit in size. Timing generation can be provided either internally or externally, depending on the designer's requirements. The Am29C668 offers a very high drive capability — one device can directly drive four 16-bit wide banks of DRAMs plus the extra bits required for error detection and correction, or two 32-bit-wide banks plus check bits, without external buffers (Figure 2). As a result, memory-control systems can now be built with a minimum number of devices. Also, minimum access times can be achieved through burst addressing and by reading data from different memory banks using bank interleaving. The four independent CAS signals of the Am29C668 can be used to control four separate banks of DRAMs, or to determine the access of a certain byte within a 32-bit word.

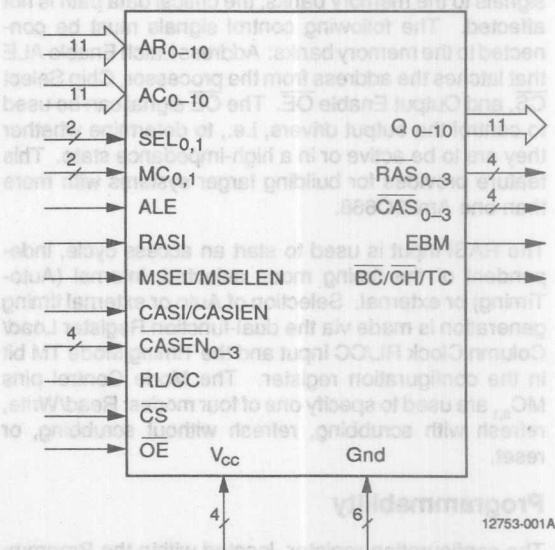


Figure 1. Am29C668 Logic Symbol

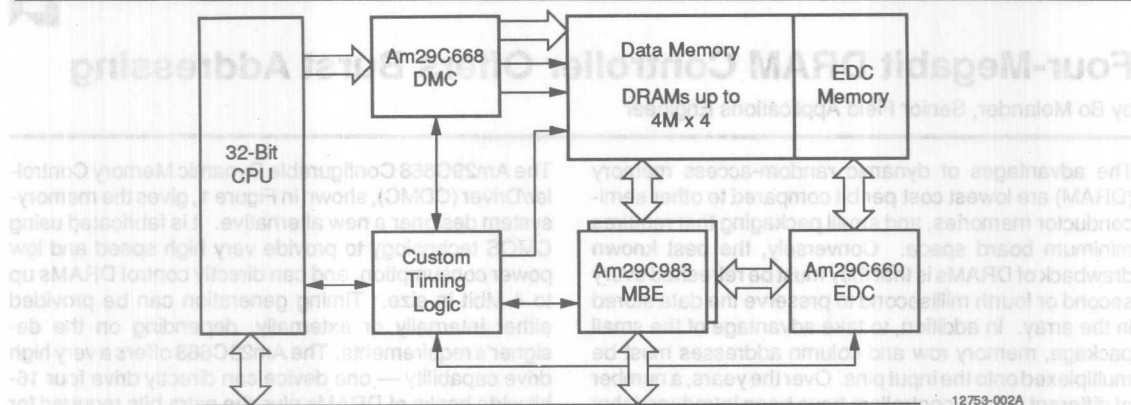


Figure 2. 32-Bit DRAM System

A BASIC SYSTEM

In a typical microprocessor system, the Column and Row Address inputs AC_{0-10} and AR_{0-10} of the Am29C668 are connected to the address bus coming from the microprocessor; the multiplexed Address outputs Q_{0-10} of the CDMC are connected to the memory address inputs. Since the CDMC only generates addresses and control signals to the memory banks, the critical data path is not affected. The following control signals must be connected to the memory banks: Address Latch Enable ALE that latches the address from the processor, Chip Select \overline{CS} , and Output Enable \overline{OE} . The \overline{OE} signal can be used to control the output drivers, i.e., to determine whether they are to be active or in a high-impedance state. This feature provides for building larger systems with more than one Am29C668.

The RAS_I input is used to start an access cycle, independent of the timing mode selected, internal (Auto-Timing) or external. Selection of Auto or external timing generation is made via the dual-function Register Load/Column Clock RL/CC input and the Timing Mode TM bit in the configuration register. The Mode Control pins $MC_{0,1}$ are used to specify one of four modes: Read/Write, refresh with scrubbing, refresh without scrubbing, or reset.

Programmability

The configuration register, located within the Programmable Registers and Logic block of the Am29C668 (Figure 3), is loaded from the column-address bus with the input signal RL/CC . It can be programmed to select a number of options including DRAM size: 64K, 256K, 1 Mbit or 4 Mbit. With this wide selection, the CDMC can be used with the mass-produced DRAMs available today, as well as with the larger DRAMs of tomorrow. The

refresh and scrubbing counters are automatically set to the correct count when the DRAM size is selected. The Column Address Strobe outputs CAS_n can be selected as bank or byte select. Also, many DRAMs support RAS -before- CAS refresh, in which a refresh counter internal to the DRAM is used to select the next row in the memory array to be refreshed. The Am29C668 can be programmed to support this refresh scheme. Normally, the row address for refreshing is taken from a counter within the CDMC. Also, burst addressing is selectable when required for use with nibble-mode or page-mode DRAMs. For nibble-mode DRAMs, burst addresses are generated for four words at a time.

Timing Generation

As mentioned earlier, internal or external timing generation can be used with the Am29C668 CDMC. Internal timing generation (auto-timing mode) offers optimum performance when used with 100-ns memories (500-pF load); options exist for controlling RAS_n and CAS_n through gating with external signals. The auto-timing mode is selected via the TM bit and the memory-access cycle starts when RAS_I is activated. The row address is immediately available at the memory address inputs; at a specific time later, the four RAS_n signals go active so that the DRAM can latch this address. The row addresses presented to the DRAM change to column addresses, controlled by the internal timing delay and gated with the Multiplexer Select $MSEL$ input, or in case of external timing generation, controlled only by $MSEL$. After a delay, the CAS_n outputs from the Am29C668 go active, signaling to the DRAM that the CAS_n addresses are stable.

The CAS timing is either generated internally in the auto-timing mode and enabled with the $CASI$ input, or

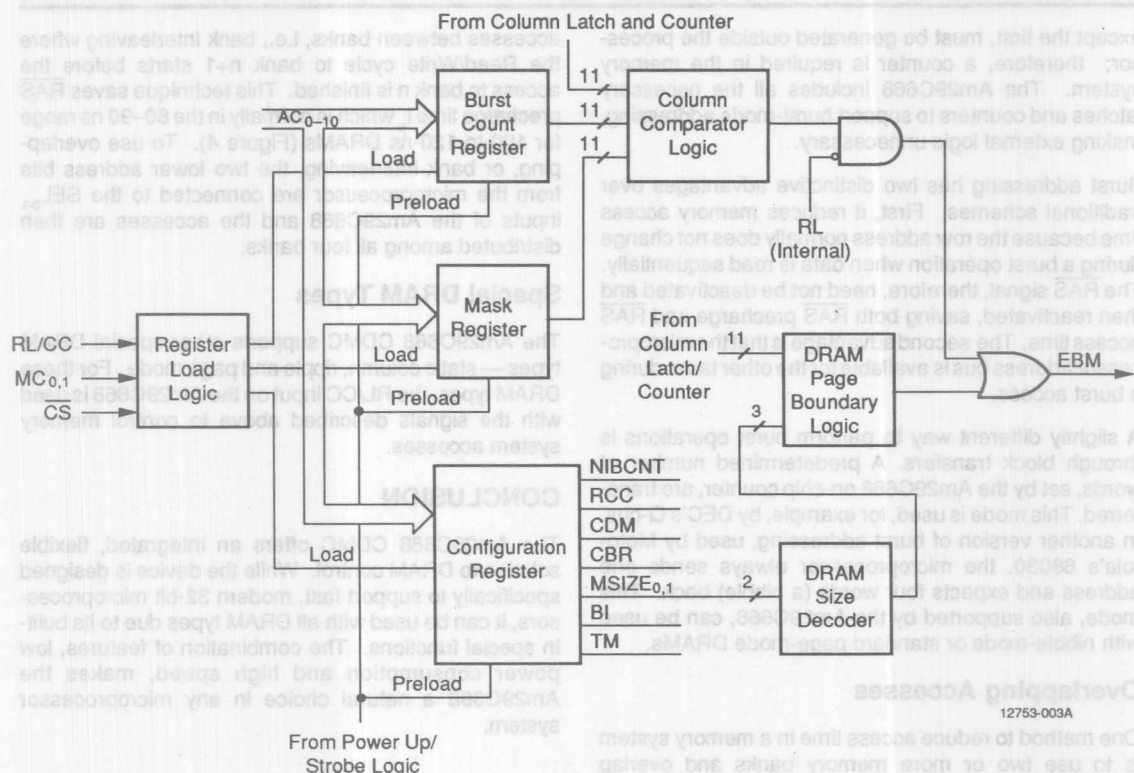


Figure 3. Programmable Registers and Logic

controlled only by the CAS_i input in the external-timing mode. Optimum timing varies with the speed of the DRAMs. Internal timing generation can be used with 100-ns memories without external drivers. The CDMC can be used with faster memories, in the 60-to-90 ns range, using delay lines or PAL devices for external timing control.

Memory-System Refresh

The Am29C668 supports three different types of refresh: normal, $\overline{\text{CAS}}$ before $\overline{\text{RAS}}$, and refresh with scrubbing. During normal refresh, the MC_{0,1} inputs initiate the refresh cycle. The internal refresh counter, automatically adjusted for the size of the DRAM, outputs a refresh address. At the same time, all four $\overline{\text{RAS}}$ signals go active, signaling to the DRAM that a refresh cycle has begun.

$\overline{\text{CAS}}$ -before- $\overline{\text{RAS}}$ refreshing, used with DRAMs that have on-chip refresh counters, is accomplished by changing the normal order that $\overline{\text{RAS}}$ _n and $\overline{\text{CAS}}$ _n are presented to the memory. This refresh support is selected by enabling the $\overline{\text{CAS}}$ -before- $\overline{\text{RAS}}$ bit in the configuration register; the Am29C668 then outputs all four

$\overline{\text{CAS}}$ _n signals, followed by the four $\overline{\text{RAS}}$ _n, indicating to the DRAM that a $\overline{\text{CAS}}$ -before- $\overline{\text{RAS}}$ refresh cycle has started.

The refresh with scrubbing mode is used when an error detection and correction (EDC) device, such as the Am29C660, is used in the system. One word is read from the memory system during a refresh, passed through the EDC and written back, completely transparent to the microprocessor. In this way, the entire memory bank can be automatically cleared from erroneous bits.

Burst Addressing

One of the most significant improvements in the Am29C668 over older-generation memory controllers is the availability of burst-addressing protocol for the memory banks. Burst addressing is used by some very fast RISC microprocessors, such as the Am29000, and in most cache-memory systems. During burst addressing, the microprocessor, or cache controller, sends only the first address when a consecutive block of data is being accessed. The memory system then latches this address and, via handshaking, sends or accepts data at address n, address n+1, n+2, etc. All the addresses,

except the first, must be generated outside the processor; therefore, a counter is required in the memory system. The Am29C668 includes all the necessary latches and counters to support burst-mode addressing, making external logic unnecessary.

Burst addressing has two distinctive advantages over traditional schemes. First, it reduces memory access time because the row address normally does not change during a burst operation when data is read sequentially. The RAS signal, therefore, need not be deactivated and then reactivated, saving both RAS precharge and RAS access time. The second advantage is that the microprocessor address bus is available for the other tasks during a burst access.

A slightly different way to perform burst operations is through block transfers. A predetermined number of words, set by the Am29C668 on-chip counter, are transferred. This mode is used, for example, by DEC's Q-bus. In another version of burst addressing, used by Motorola's 68030, the microprocessor always sends one address and expects four words (a nibble) back. This mode, also supported by the Am29C668, can be used with nibble-mode or standard page-mode DRAMs.

Overlapping Accesses

One method to reduce access time in a memory system is to use two or more memory banks and overlap

accesses between banks, i.e., bank interleaving where the Read/Write cycle to bank $n+1$ starts before the access to bank n is finished. This technique saves RAS precharge time t_p , which is normally in the 80–90 ns range for 100-to-120-ns DRAMs (Figure 4). To use overlapping, or bank interleaving, the two lower address bits from the microprocessor are connected to the $SEL_{0,1}$ inputs of the Am29C668 and the accesses are then distributed among all four banks.

Special DRAM Types

The Am29C668 CDMC supports other special DRAM types — static column, ripple and page mode. For these DRAM types, the RL/CC input on the Am29C668 is used with the signals described above to control memory system accesses.

CONCLUSION

The Am29C668 CDMC offers an integrated, flexible solution to DRAM control. While the device is designed specifically to support fast, modern 32-bit microprocessors, it can be used with all DRAM types due to its built-in special functions. The combination of features, low power consumption and high speed, makes the Am29C668 a natural choice in any microprocessor system.

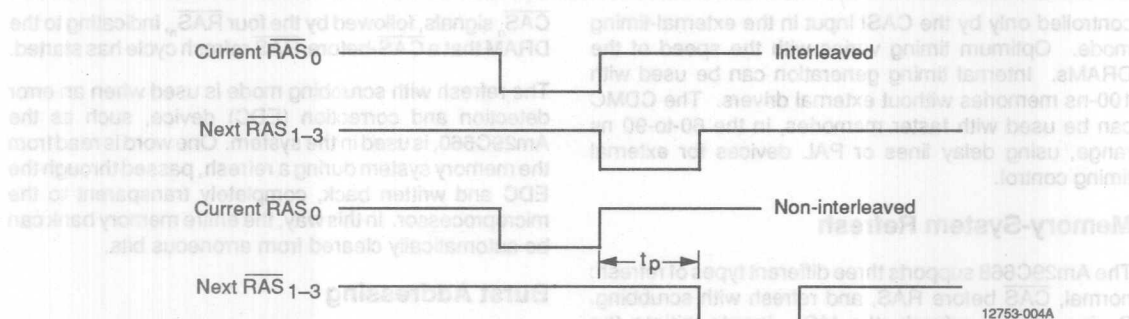


Figure 4. Am29C668 Enhancement Bank-Interleave Feature

High-Speed VCEP Demonstration Board Using the Am29C668 Configurable Dynamic Memory Controller

by Vineet Dujari, Applications Manager

VCEP OVERVIEW

The Am95C71 Video Data Compression/Expansion Processor (VCEP), Figure 1, performs CCITT(T.4 and T.6)-compatible video-data processing at very high speeds for applications in low-cost real-time document storage and retrieval systems.

Using patented hardware-based compression/expansion techniques, the VCEP features throughput averaging 60 to 80 Mbit/s, i.e., over six pages per second, for CCITT standard documents. It has a dual-bus interface for source and destination memories; however, the VCEP can easily be used in low-cost single-bus systems. A simple register-based interface, controlled by any microprocessor, is used to program the device. A set of six registers contain parameters such as page width, page length, and the required coding algorithm. Normally, these registers are written only once during initialization; the device is controlled during normal operation by using the command/status register.

The hardware interface for the VCEP is straightforward. The device registers are accessed using standard chip-select and Read/Write control signals. Two on-chip

input(source) and output(destination) FIFOs, 16 locations deep, provide for burst-mode data transfer from external memory. When the VCEP is enabled, I/O data-request signals indicate which data FIFO needs service and external control logic activates the appropriate data strobe. The FIFOs can also be accessed, using the register interface, in a low-end system to avoid additional control logic.

DEMONSTRATION BOARD DESCRIPTION

The VCEP demonstration board, Figure 2, plugs into a PC-AT* slot and drives an NEC multisync video monitor. The device operates in a dual-bus configuration in this application; compressed-image data is fed onto the board from the PC-AT bus and the expanded-image data is retrieved via the image-data bus. Since the VCEP is very fast, it requires no frame buffer to hold the expanded image; it can drive the video monitor directly. It performs image expansion in real time, 60 times a second, to display a flicker-free image. Alternatively, the external logic can provide a different coded image for each frame and an animation sequence can be displayed.

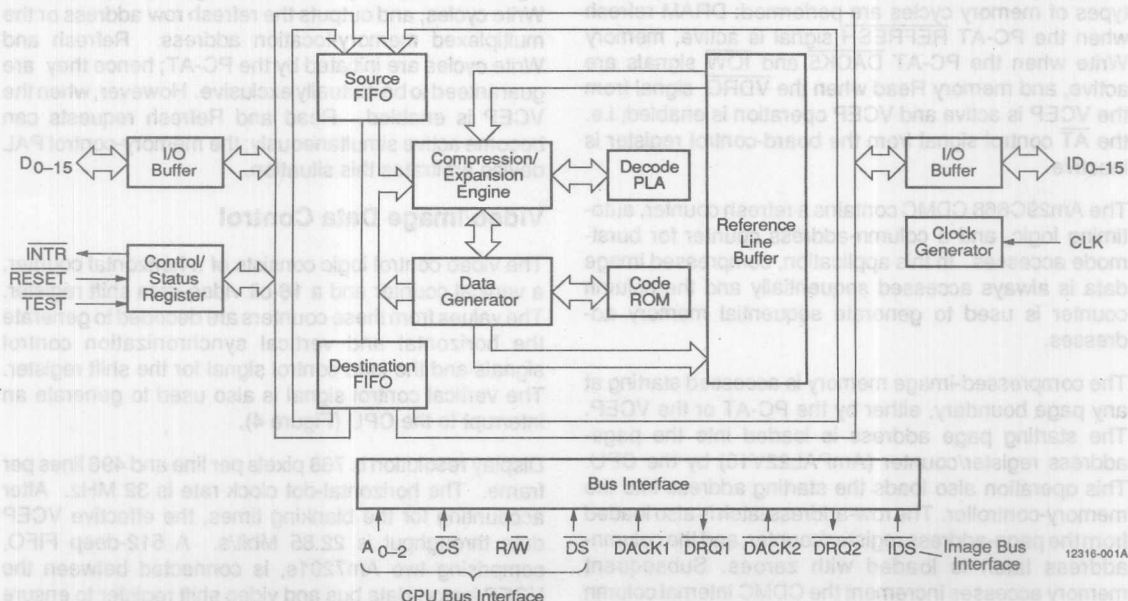


Figure 1. VCEP Block Diagram

* PC-AT is a registered trademark of IBM Incorporated.

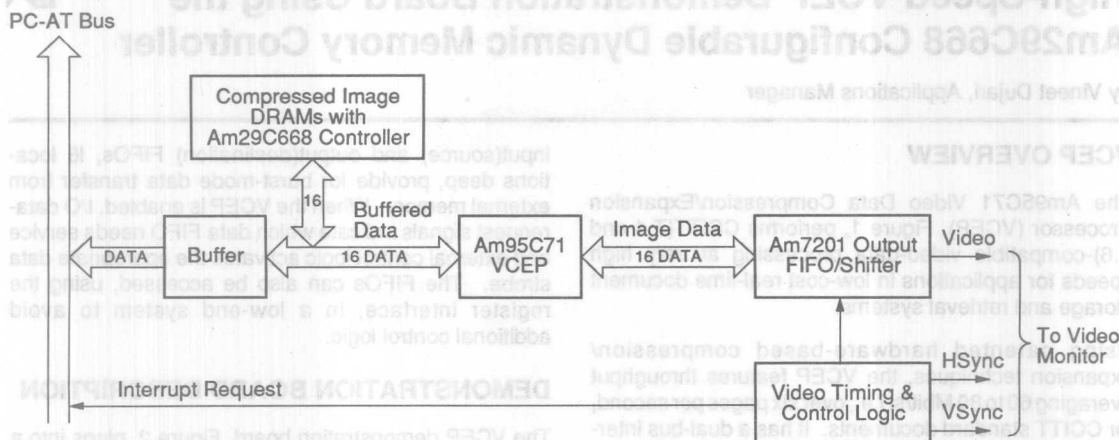


Figure 2. VCEP Demonstration Board

Compressed Image Memory

This board requires a dedicated memory buffer to hold the compressed images because the bandwidth of the PC-AT bus is not fast enough to supply the image data at the required rate. However, in other systems, the main memory could be used to hold the coded-image data.

The compressed-image memory is designed using DRAMs; the array is controlled by an Am29C668 Configurable Dynamic Memory Controller (CDMC). Three types of memory cycles are performed: DRAM refresh when the PC-AT REFRESH signal is active, memory Write when the PC-AT DACK5 and IOW signals are active, and memory Read when the VDRQ signal from the VCEP is active and VCEP operation is enabled, i.e. the AT control signal from the board-control register is inactive.

The Am29C668 CDMC contains a refresh counter, auto-timing logic, and a column-address counter for burst-mode accesses. In this application, compressed image data is always accessed sequentially and the column counter is used to generate sequential memory addresses.

The compressed-image memory is accessed starting at any page boundary, either by the PC-AT or the VCEP. The starting page address is loaded into the page-address register/counter (AmPAL22V10) by the CPU. This operation also loads the starting address into the memory-controller. The row-address latch is also loaded from the page-address register/counter, and the column-address latch is loaded with zeroes. Subsequent memory accesses increment the CDMC internal column counter, using the CAS0 signal fed into the Am29C668 via the column-counter control signal RL/CC. When a page boundary is reached, the End Burst Mode EBM

signal becomes active. This causes the external page-address register/counter to be incremented, and the starting address of the next page is loaded into the Am29C668 address latch.

Figure 3 shows the details of the PC-AT interface. The memory-control PAL[®] device is used to generate the RAS-Input RASI and Mode-Control signals MC₁₋₀ for the CDMC. The MC₁₋₀ specify one of four modes of operation of the Am29C668. Depending on the specific signals, the CDMC generates timing for refresh or Read/Write cycles, and outputs the refresh row address or the multiplexed memory-location address. Refresh and Write cycles are initiated by the PC-AT; hence they are guaranteed to be mutually exclusive. However, when the VCEP is enabled, Read and Refresh requests can become active simultaneously; the memory-control PAL device arbitrates this situation.

Video/Image Data Control

The video control logic consists of a horizontal counter, a vertical counter and a 16-bit video-data shift register. The values from these counters are decoded to generate the horizontal and vertical synchronization control signals and the load control signal for the shift register. The vertical control signal is also used to generate an interrupt to the CPU (Figure 4).

Display resolution is 768 pixels per line and 496 lines per frame. The horizontal-dot clock rate is 32 MHz. After accounting for the blanking times, the effective VCEP data throughput is 22.85 Mbit/s. A 512-deep FIFO, comprising two Am7201s, is connected between the VCEP image-data bus and video shift register to ensure that the image data is available even when the compression ratio in some local area of the document is not too good.

The image-data control PAL device transfers a word from the VCEP to the external FIFO when the Image Data Request $\overline{\text{IDRQ}}$ signal becomes active and FIFO-Full $\overline{\text{FF}}$ signal is inactive. This expanded image is then loaded into the video shift register and shifted out.

CONCLUSION

This demonstration board illustrates the high-speed operation of the VCEP and the Am29C668 CDMC. However, as mentioned before, the VCEP is capable of much higher throughput: 70 to 80 Mbit/s vs 23 Mbit/s

required by the multisync monitor. Throughput rates vary depending on image complexity and coding choice. Running at the maximum clock rate of 20 MHz, the Am95C71 typically handles MMR coding at 65–75 Mbit/s, MR coding ($k = 4$) at 70–80 Mbit/s, and MH coding at 75–85 Mbit/s.

The VCEP can be designed to perform high-speed image compression/expansion in document-storage and retrieval systems. Using this device, retrieval systems can store large amounts of compressed data that can be quickly retrieved in real time.

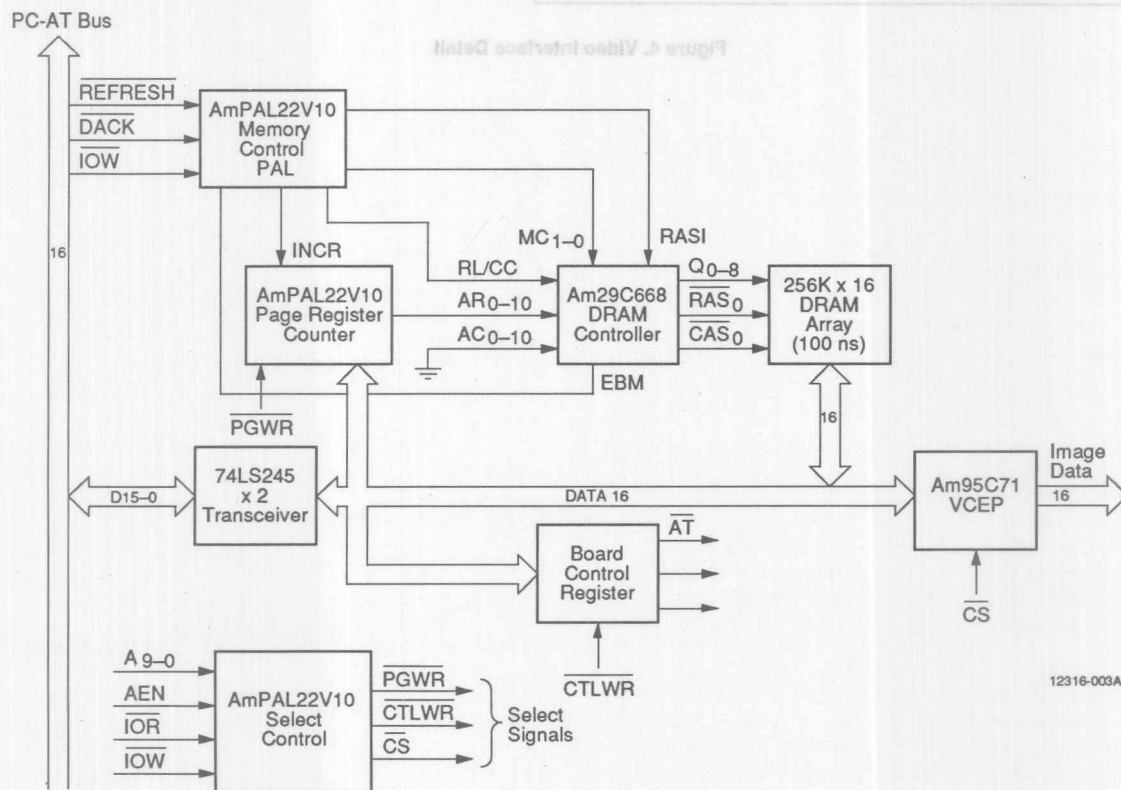


Figure 3. PC-AT Interface

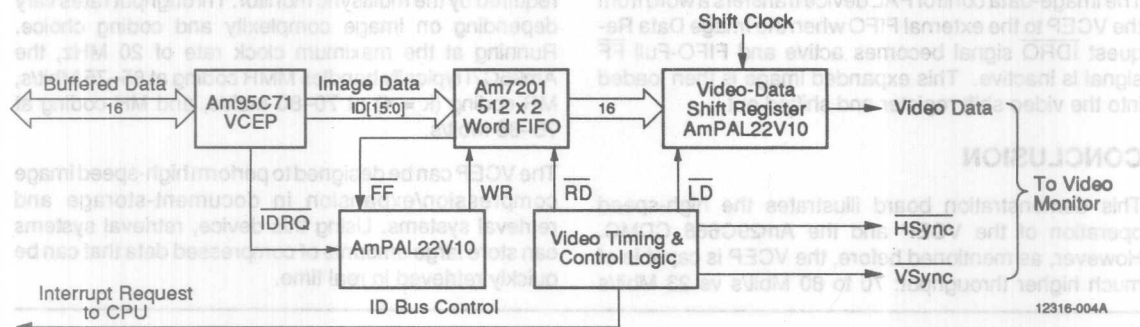


Figure 4. Video Interface Detail

Universal Coprocessor Platform (UCP) and Network Terminator-S Interface (NTs)



INTRODUCTION

The following is selected section of the Universal Coprocessor Platform (UCP) and Network Terminator-S Interface (NTs) technical manual which is applicable to the Am29C668 and MBE Am29C983. For further information please see the UCP and NTs Technical Manual, REV 1 (PID #14748A). The technical manual is available through the PCD division product line marketing.

The UCP is a full length IBM-PC I/O profiles card that plugs into an XT or AT compatible personal computer, and was designed to meet the following objectives:

1. To provide sufficient co-processing power to run ISDN software pertinent to layer 3 and below.
2. To provide a demonstration/application and evaluation tool for AMD's family of ISDN/Telecom components.
3. To realize a combination of AMD hardware and software that can be certified as being compliant with ISDN compatible switching systems.
4. To provide a development platform for a range of PC based applications, including terminal adapters, network terminators and X.25.
5. To provide an environment for the demonstration and evaluation of AMD S/U transmission performance.

In order to accommodate the objectives within the scope of their applications, the UCP architecture was designed to meet two primary goals: high performance and expandability. Figure 1 shows a block diagram of the UCP.

The goal for high performance stems from two facts: 1) the UCP's processor will be operating under a multi-tasking operation system, which accounts for added

overhead in protocol processing, and 2) the simultaneous use of both B channels at 64KBps and D channel at 16KBps of the ISDN requires that the local processor possess sufficient processing power and memory storage in order to efficiently process data through multiple layers of communications protocols.

The goal for expandability stems from the requirement of providing a development/evaluation platform which can accommodate a number of hardware and software applications. This provides a cost effective tool for supporting future hardware/software applications. To this end, the UCP is designed around a set of cards, or ISDN Personality Modules (IPM), implement the specific hardware functions of the application while the motherboard (UCP) implements the software/system functions. The key advantage to this approach is that it provides a cost effective development/application platform which can easily be modified to accommodate other PC connectivity applications. This decoupling ensures that enhancements by OEMs have localized impact on the UCP architecture.

Dynamic RAM Subsystem

Two 100ns 256K by 8-bit dynamic RAM modules implement the UCP's dynamic memory. They are arranged as two banks, U39A and U39B, where U39B is the even byte and U39A the odd byte. Data to/from the dynamic RAM is transferred over the local processor bus, LAD15:0. BAO (U25) in conjunction with LBHE (U27) determine whether the even or odd bank, or both, are accessed during the cycle. Dynamic RAM cycles are controlled by the Am29C668 dynamic RAM controller (U30) in conjunction with a synchronous Moore based state machine (U21) and all dynamic RAM cycles are synchronous to the local processor clock, LCLK.



Figure 1. UCP Block Diagram

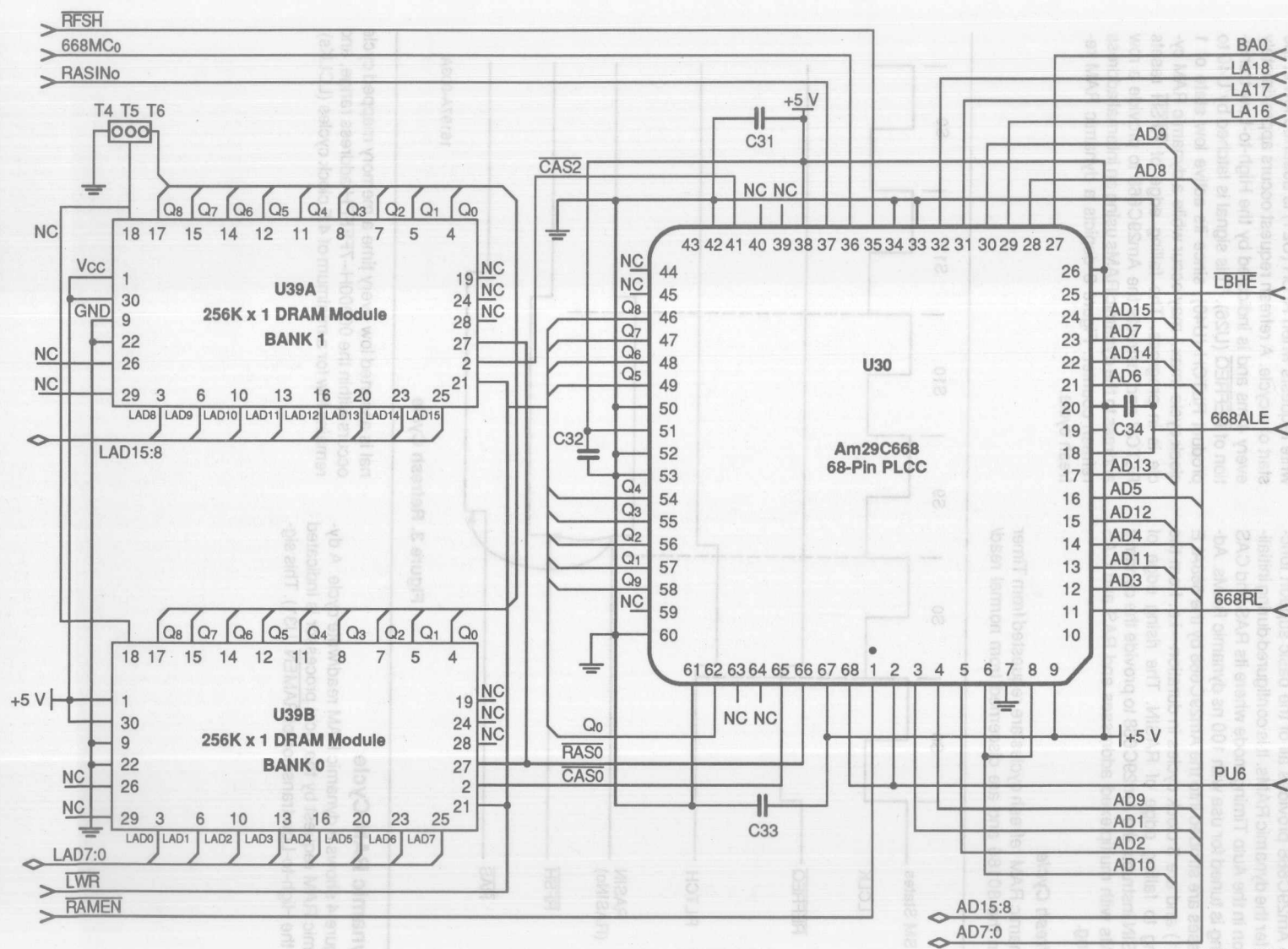


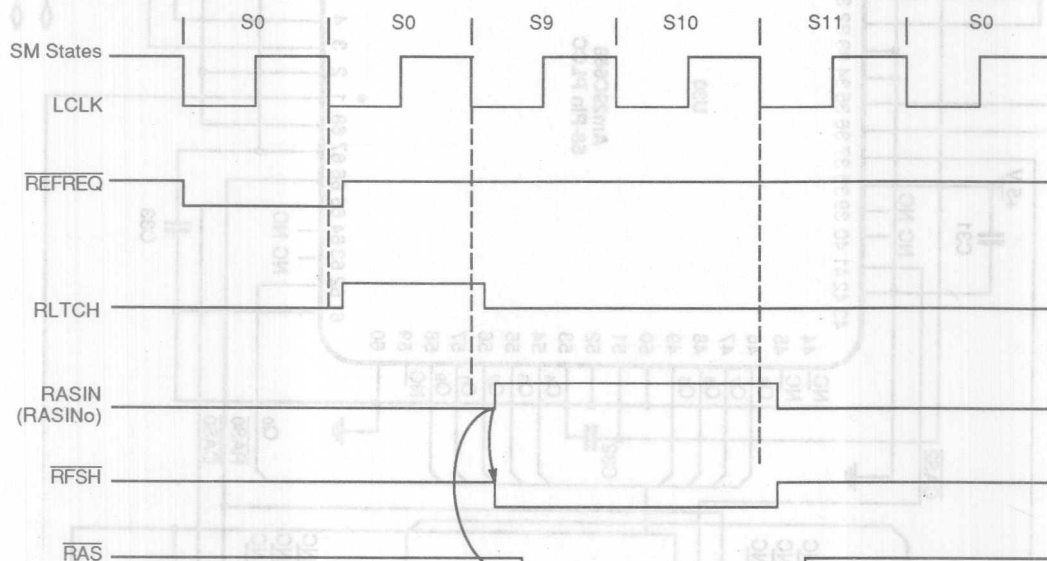
Figure 2. DRAM Controller/SIMM Interface

The Am29C668 provides all of the basic support function for the dynamic RAMs. It is configured during initialization in the Auto Timing mode where its RAS and CAS timing is tuned for use with 100 ns dynamic RAMs. Addresses are strobed into the Am29C668 by the 668ALE (U21) and are 3 clock cycles in duration—i.e. from the rising to falling edge of RASIN. The rising edge of RASIN instructs the Am29C668 to provide the dynamic RAMs with multiplexed addresses and RAS and CAS timing.

Refresh Cycle

Dynamic RAM refresh cycles are requested from Timer 1 on the 80186 and are discerned from normal read/

write requests when $\overline{\text{RFSH}}$ (U21) is asserted low at the start of a cycle. A refresh request occurs approximately every 4 ms and is indicated by the High-to-Low transition of $\overline{\text{REFREQ}}$ (U26). This signal is latched by U40 to product RLTC (U40), since its active low state of 1 clock cycle (max.) may occur while a dynamic RAM cycle is in progress. The falling edge of $\overline{\text{RFSH}}$ resets RLTC and instructs the Am29C668 to provide a row address to the dynamic RAMs using an internal address refresh counter. Figure 3 depicts a dynamic RAM refresh cycle.



16197A-003A

Figure 3. Refresh Cycle

Dynamic RAM Cycle

Figure 4 shows a dynamic RAM read/write cycle. A dynamic RAM request by the local processor is indicated by the High-to-Low transition of $\overline{\text{RAMEN}}$ (U31). This sig-

nal is asserted low every time a memory mapped cycle occurs within the 00000H–7FFFFH address range, and remains low for a minimum of 4.5 clock cycles (LCLKs).

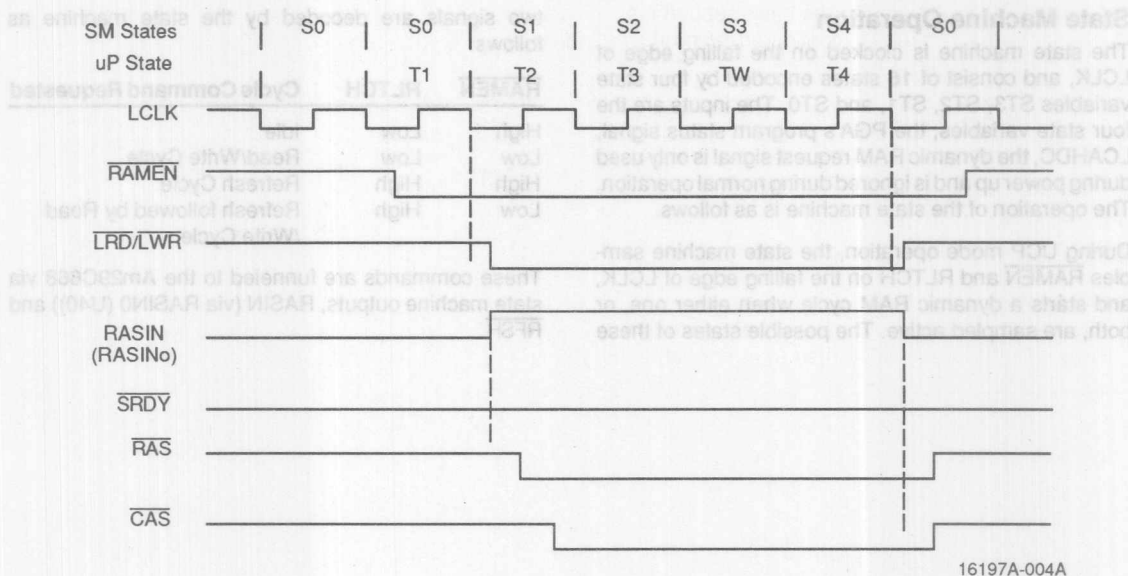


Figure 4. Dynamic RAM READ/WRITE Cycle

Refresh – Read/Write Dynamic RAM Cycle

Figure 5 depicts the case where a refresh and read/write request are detected active simultaneously. Since $\overline{\text{RAMEN}}$ and $\overline{\text{RLTCH}}$ occur asynchronously to each other, the state machine resolves contention between

$\overline{\text{RAMEN}}$ and $\overline{\text{RLTCH}}$ by requesting wait states from the local processor via $\overline{\text{SRDY}}$. $\overline{\text{SRDY}}$ is asserted low when both signals are sampled active simultaneously or when $\overline{\text{RAMEN}}$ goes active during the execution of a refresh cycle. The maximum number of wait states inserted is 4.

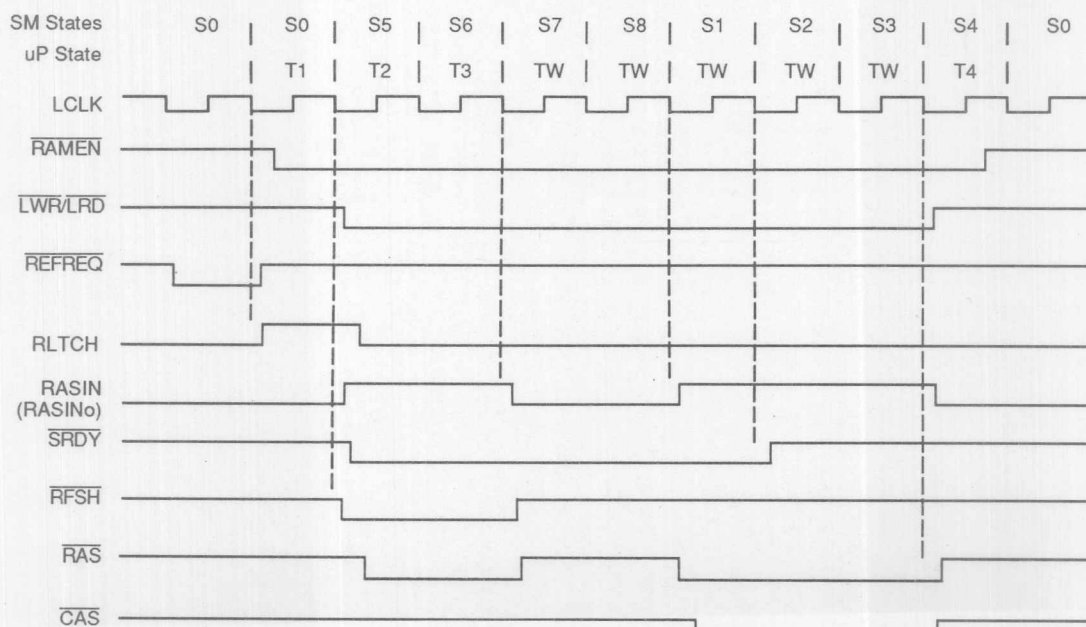


Figure 5. Refresh Followed by READ/WRITE Cycle



State Machine Operation

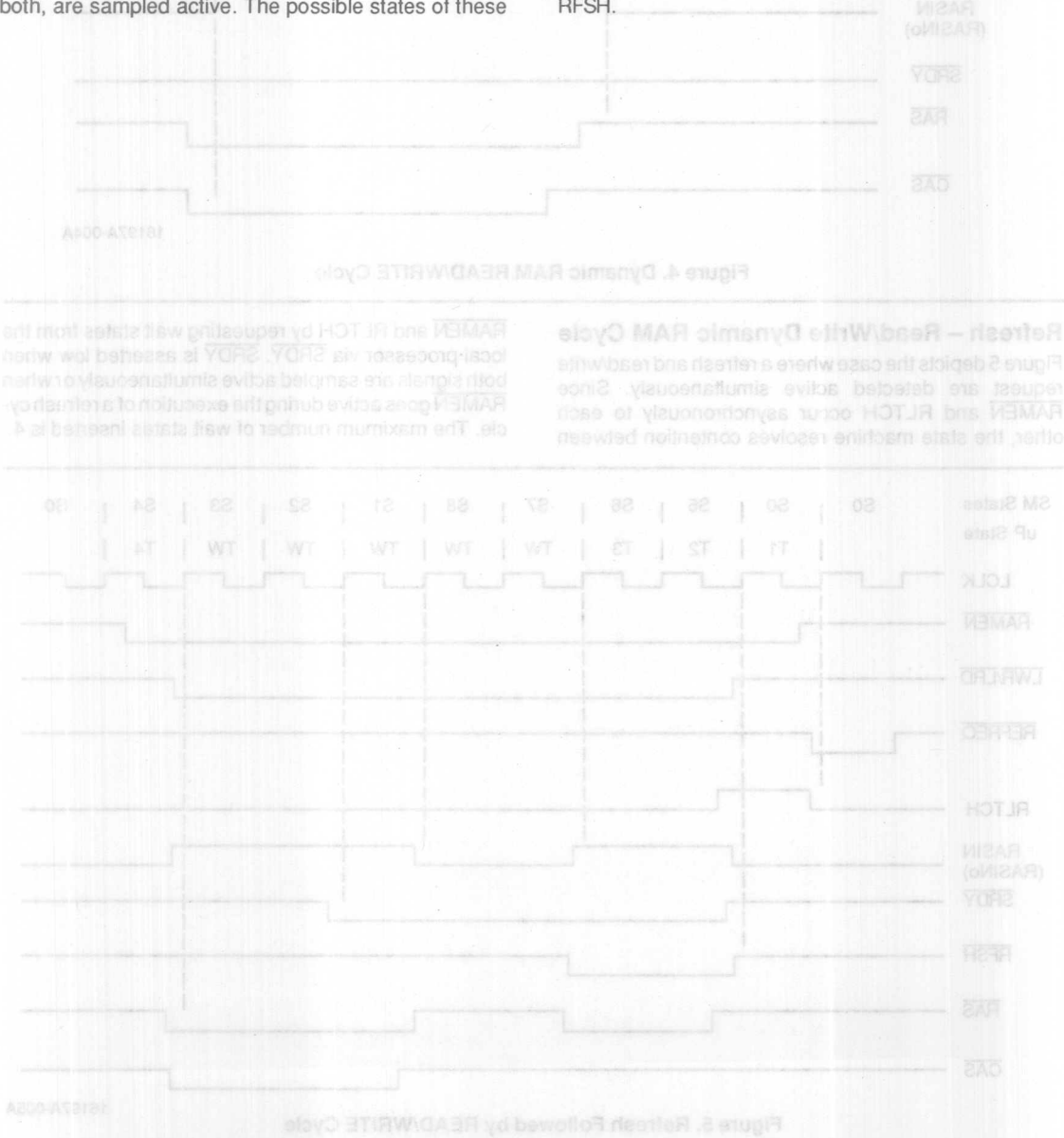
The state machine is clocked on the falling edge of LCLK, and consist of 16 states encoded by four state variables ST3, ST2, ST1, and ST0. The inputs are the four state variables, the PGA's program status signal, LCAHDC, the dynamic RAM request signal is only used during power up and is ignored during normal operation. The operation of the state machine is as follows.

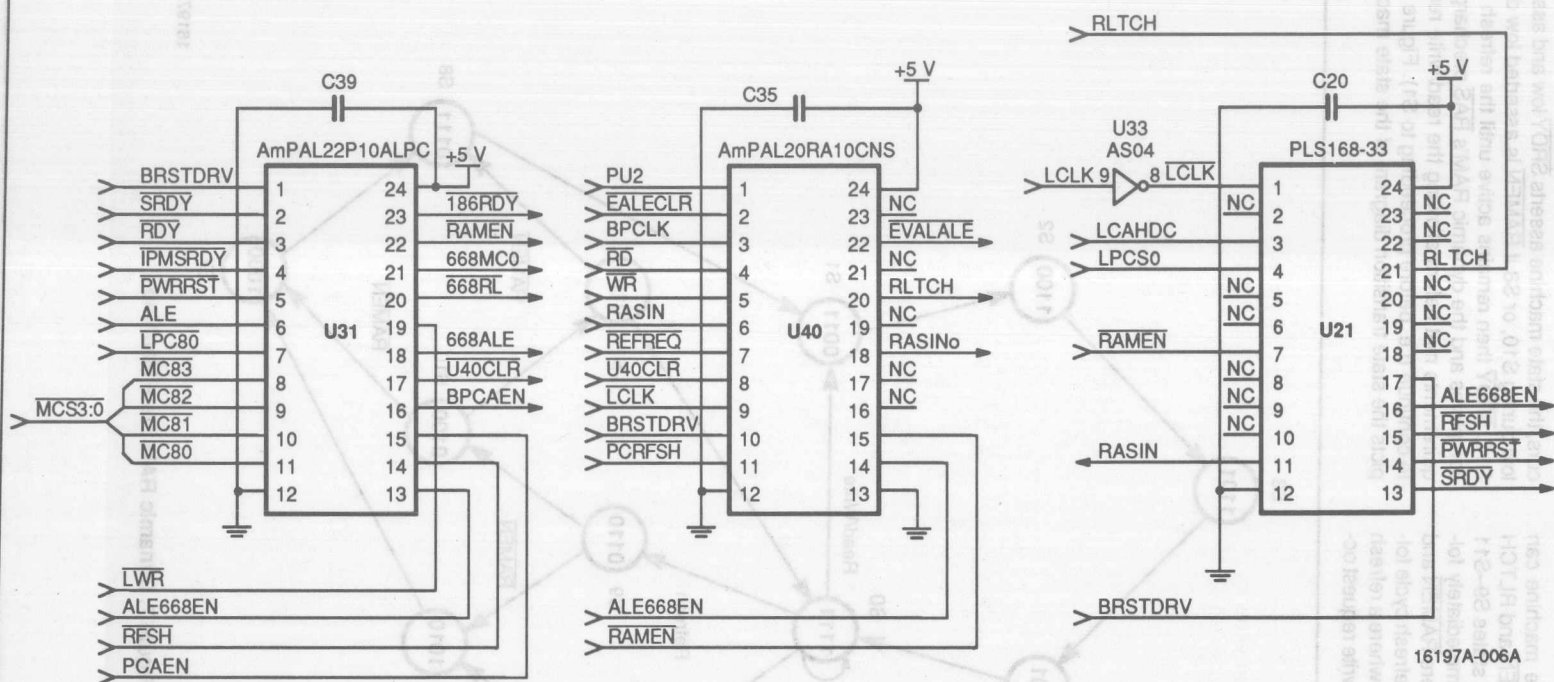
During UCP mode operation, the state machine samples RAMEN and RLTCR on the falling edge of LCLK, and starts a dynamic RAM cycle when either one, or both, are sampled active. The possible states of these

two signals are decoded by the state machine as follows:

RAMEN	RLTCH	Cycle Command Requested
High	Low	Idle
Low	Low	Read/Write Cycle
High	High	Refresh Cycle
Low	High	Refresh followed by Read /Write Cycle

These commands are funneled to the Am29C668 via state machine outputs, RASIN (via RASIN0 (U40)) and RFSH.





16197A-006A

Figure 6. DRAM State Machine (U21)/Memory Decode (U31)/DRAM Refresh Control (U31/U40)

The three possible paths that the state machine can traverse based on the sampling of $\overline{\text{RAMEN}}$ and RLTCH are: states S1–S4 for read/write cycles, states S9–S11 for refresh cycles, and states S5–S8 immediately followed by states S1–S4 for the case where $\overline{\text{RAMEN}}$ and RLTCH are active simultaneously (i.e. refresh cycle followed by a read/write cycle). In the case where a refresh cycle is already in progress and a read/write request oc-

curs, the state machine asserts $\overline{\text{SRDY}}$ low and asserted low during S10, or S8 if $\overline{\text{RAMEN}}$ is asserted low during S11. $\overline{\text{SRDY}}$ then remains active until the refresh cycle completes and the dynamic RAM's RAS precharge requirements met before allowing the read/write request to continue (i.e. before proceeding to S1). Figure 7 depicts the state transition diagram of the state machine.

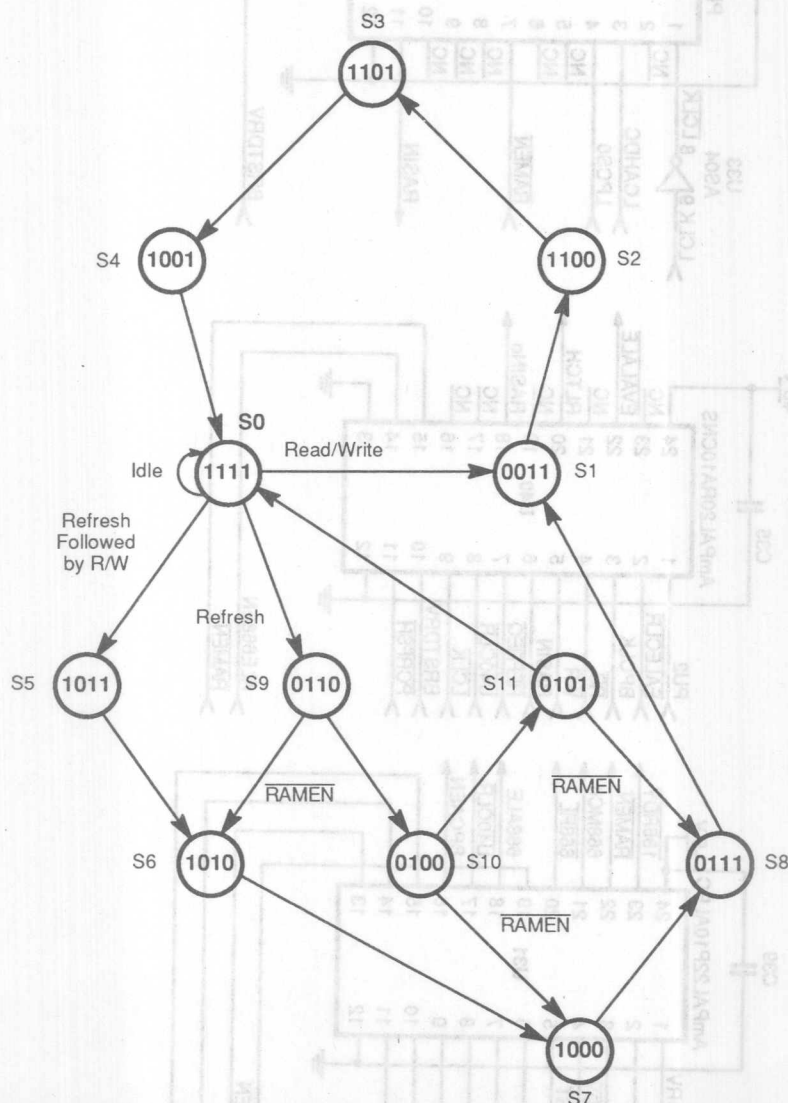


Figure 7. Dynamic RAM State Machine

16197A-007A

Am29C668 Dynamic RAM Controller

The Am29C668 provides the address multiplexing and timing control signals (\overline{RAS} , \overline{CAS}) to the UCP's dynamic RAM modules (MSC2329YS3/KS3). The UCP program code initializes it for a memory size of 256K (2 banks) and configures it to operate in the Auto-Timing Mode. A state machine (U21), implemented with a programmable logic sequencer (PLS168-33), is used for managing dynamic RAM cycles and the Am29C668's power-up initialization sequence.

A sequence of three operations must occur upon power-up in order for the Am29C668 to function properly. These three operations, Reset, Wake Up and Register Load, are encoded via the Am29C668's MC0, MC1, and RL pins, and must occur without any transitions on its ALE or \overline{CS} (except \overline{CS} for Register Load operation) pins. To this end, the state machine outputs ALE668EN and \overline{PWRST} , in conjunction with \overline{RASIN} (U21), 668MC0 (U31), LCAHDC (U25), and $\overline{LPCS0}$ (U26), are used to control the operation sequencing during power-up as follows.

Immediately upon power-up the state machine enters the Idle state, S0, and samples LCAHDC from the PGA. If it samples this signal high, it enters the Reset opera-

tion encompassing states S15 and S14. Here, \overline{PWRST} is asserted low during both states and \overline{RASIN} (U21) is asserted high during S15 and low during S14. It then proceeds to the Wake Up operation encompassing state S13 and S12 where, upon entering S13, \overline{RFSH} is asserted low, and remains low, and \overline{RASIN} is asserted high then toggled low during S12. The state machine then toggles between these two states until LCAHDC is sampled low. When LCAHDC is sampled low, the state set \overline{RFSH} high and \overline{RASIN} low and re-enters the Idle state, S0. \overline{RASIN} and \overline{RFSH} then remain in these states until either a refresh or read/write request occurs. Since the PGA requires a minimum of 50 ms of program time, the Am29C668 requirement of 5 \overline{RAS} cycles (minimum) during the Wake Up operation is easily met.

The Register Load operation is handled by the UCP's program code. When the write cycle to the Am29C668's configuration register occurs via $\overline{LPCS0}$, 668MC0 is set high and 668RL follows $\overline{LPCS0}$. The Low-to-High transition of 668RL resets 668MC0 low, latches the configuration data into the Am29C668's register, and sets ALE668EN high. Thus enabling ALEs to the Am29C668 via 668ALE (U31). Figure 8 shows the state transition diagram of the state machine during-up.

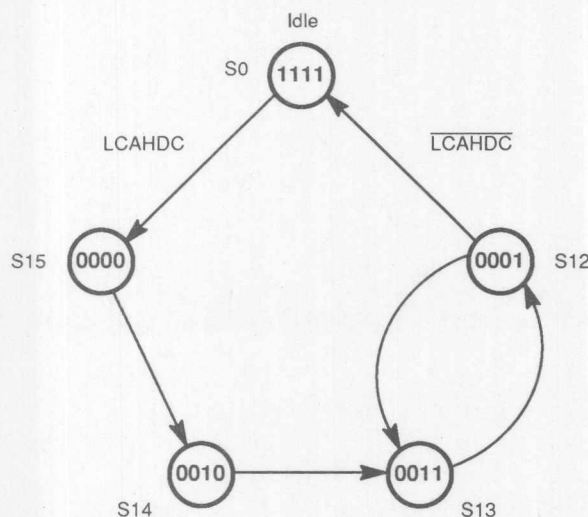


Figure 8. State Machine Initialization State Diagram

16197A-008A

Shared Memory Address Multiplexing

The address which is placed on the shared memory address bus is controlled by $\overline{HREQ0}$ and LWAIT. LWAIT controls two 74ALS646 (U10 and U11) transceivers via 646GEN from U18, and $\overline{HREQ0}$ controls two 74ALS244 buffers via \overline{HABE} from U25. The addresses are avail-

able on the shared memory bus for as long as LWAIT and $\overline{HREQ0}$ are low.

During UCP mode operation the direction of the 74ALS646 transceivers is from the 80186 to shared memory and 80186 addresses are latched into U10 and

U11 by the Low-to-High transition of 646CAB (U25). The signal, 646CAB, is generated by the PGA and is the inverted sense of the 80186's ALE signal. PC addresses don't require latching and will be presented at the outputs of U8 and U9 when HREQ0 is driven low by the shared memory controller.

in EVAL mode the direction of the 74ALS646 transceivers is reversed and HREQ0 is held low. The PC address is then multiplexed onto the IPM multiplexed address/data bus via 646GEN (U18) and EVALALE. (U40). The signal, EVALALE, is the inverted sense of the PC's ALE signal extended by half a PC clock (BPCLK) to provide sufficient address hold time to the IPM.

Shared Memory Data Switching

The block of shared memory is viewed by the PC as being organized as 32K by 8-bit bytes, whereas the 80186 views this same memory as being organized as 16K by 16-bit word. The UCP uses two 32K by 8 static RAM devices, U12 and U13, to act as shared memory and views U12 as the even byte and U13 as the odd byte during single byte fetches from shared memory. Even byte accesses occur over the least significant bus, AD0 through

AD8, and odd byte accesses occur over the most significant bus, AD8 through AD15 while word fetches occur over both busses.

On the PC side all accesses are restricted to 8-bit accesses to maintain compatibility between the PC-XT and PC-AT. Data transfers to/from the UCP occur over the PC data bus—PCD0 through PCD7.

The UCP hardware managing the data path switching between these busses consists of the Am29C983 (Multiple Bus Exchange unit (MBE), and two Am22P10 PALs (U17 and U18).

The MBE incorporates four 9-bit bi-directional ports (A, B, C and D) with input and output latching capabilities. UCP MBE port assignments are as follows: port A (SRAM odd byte), port C interfaces directly to the least significant address/data bus 74ALS245 transceiver (U16) provides the interface to the most significant address/data bus of the 80186 from U12.

Two combinatorial PALs (U17 and U18) implement the port-to-port connection control of the MBE's internal pathways.

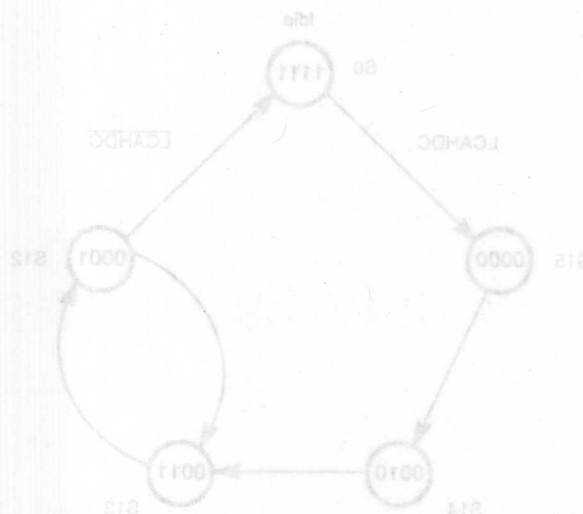


Figure 8: State Machine Initialization State Diagram

side on the shared memory bus for as long as LWAIT and HREQ0 are low.

During UCP mode operation the direction of the 74ALS646 transceivers is from the 80186 to shared memory and 80186 addresses are latched into U10 and

Shared Memory Address Multiplexing

The address which is placed on the shared memory bus is controlled by HREQ0 and LWAIT. LWAIT controls two 74ALS646 (U10 and U11) transceivers via 646GEN from U18 and HREQ0 controls two 74ALS245 buffers via HABE from U25. The addresses are avail-

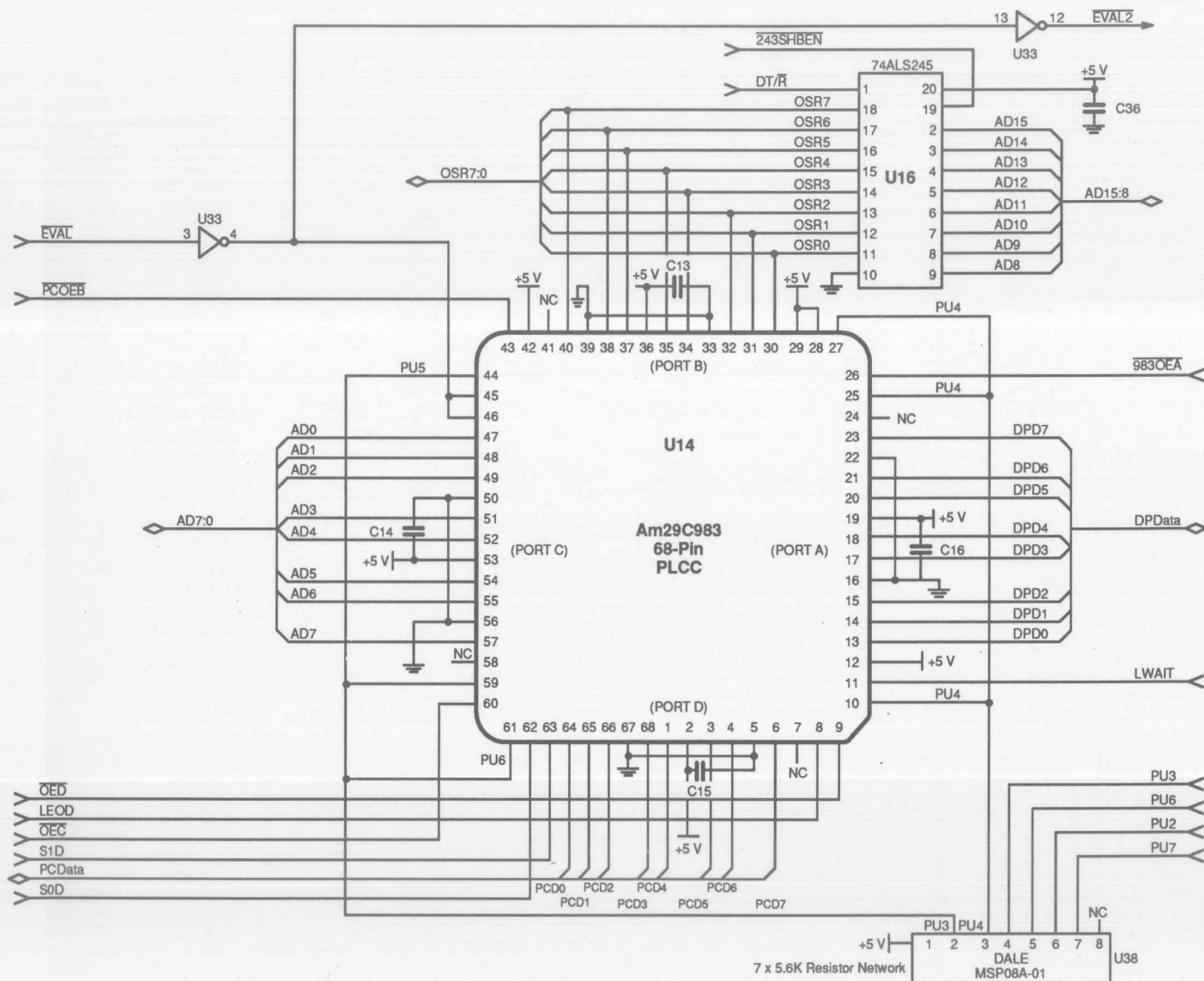


Figure 9. Shared Memory Data Path Busing

16197A-009A



CHAPTER 6

Product Data Sheets

Am2965/66 Octal Dynamic Memory Drivers with Three-State Outputs	6-3
Am29C60A CMOS Cascadable 16-Bit Error Detection and Correction Unit	6-12
Am29C660/A/B/C/D/E CMOS Cascadable 32-Bit Error Detection and Correction Circuit	6-29
Am29C668 4M Configurable Dynamic Memory Controller/Driver	6-69
Am29C676 11-Bit DRAM Driver	6-112
Am29C827A/828A High-Performance 10-Bit CMOS Bus Buffers	6-133
Am29C983A 9-Bit x 4-Port Multiple Bus Exchange	6-143
Am29C985 9-Bit x 4-Port Multiple Bus Exchange with Parity	6-156



CHAPTER 6
Product Data Sheets

6-3	Am2885 88-Qbit Dynamic Memory Drivers with Three-State Outputs
6-12	Am2880A CMOS Cascadable 18-Bit Error Detection and Correction Unit
	Am2880A/CADIE CMOS Cascadable 32-Bit Error Detection and
6-29	Correction Circuit
6-38	Am2888 4M Configurable Dynamic Memory Controller/Driver
6-112	Am2887B 11-Bit DRAM Driver
6-133	Am2887A828A High-Performance 10-Bit CMOS Bus Buffers
6-143	Am2883A 8-Bit x 4-Port Multiple Bus Exchange
6-158	Am2885 8-Bit x 4-Port Multiple Bus Exchange with Parity

Am2965/Am2966

Octal Dynamic Memory Drivers with Three-State Outputs



Advanced
Micro
Devices

DISTINCTIVE CHARACTERISTICS

- **Controlled rise and fall characteristics**
Internal resistors provide symmetrical drive to HIGH and LOW states, eliminating need for external series resistor.
- **Output swings designed to drive 16K and 64K RAMs**
 V_{OH} guaranteed at $V_{CC} - 1.15V$. Undershoot going LOW guaranteed at less than 0.5V.
- **Large capacitive drive capability**
35mA min source or sink current at 2.0V. Propagation delays specified for 50pF and 500pF loads.
- **Pin-compatible with 'S240 and 'S244**
Non-inverting Am2966 replaces 74S244; inverting Am2965 replaces 74S240. Faster than 'S240/244 under equivalent load.
- **No-glitch outputs**
Outputs forced into OFF state during power up and down. No glitch coming out of three-state.

GENERAL DESCRIPTION

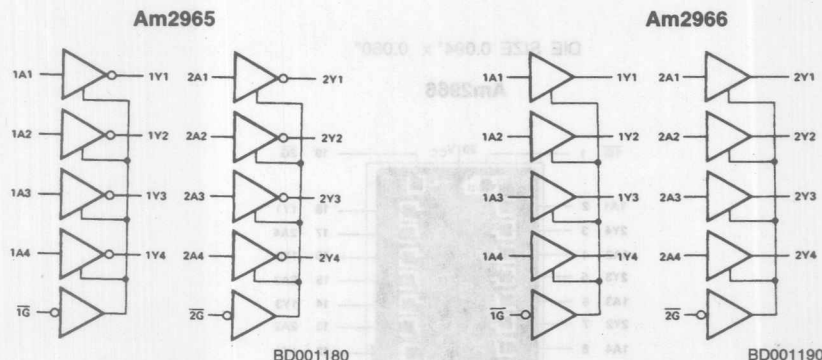
The Am2965 and Am2966 are designed and specified to drive the capacitive input characteristics of the address and control lines of MOS dynamic RAMs. The unique design of the lower output driver includes a collector resistor to control undershoot on the HIGH-to-LOW transition. The upper output driver pulls up to $V_{CC} - 1.15V$ to be compatible with MOS memory and is designed to have a rise time symmetrical with the lower output's controlled fall time. This allows optimization of Dynamic RAM performance.

The Am2965 and Am2966 are pin-compatible with the popular 'S240 and 'S244 with identical 3-state output enable controls. The Am2965 has inverting drivers and the Am2966 has non-inverting drivers.

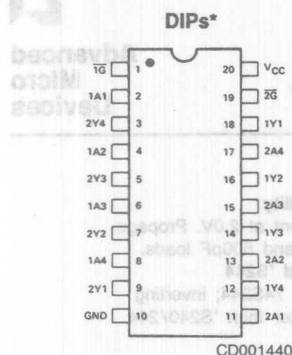
The inclusion of an internal resistor in the lower output driver eliminates the requirement for an external series resistor, therefore reducing package count and the board area required. The internal resistor controls the output fall and undershoot without slowing the output rise.

These devices are designed for use with the Am2964 Dynamic Memory Controller where large dynamic memories with highly capacitive input lines require additional buffering. Driving eight address lines or four \overline{RAS} and four \overline{CAS} lines with drivers on the same silicon chip also provides a significant performance advantage by minimizing skew between drivers. Each device has specified skew between drivers to improve the memory access worst case timing over the min and max t_{PD} difference of unspecified devices.

BLOCK DIAGRAM



CONNECTION DIAGRAM Top View



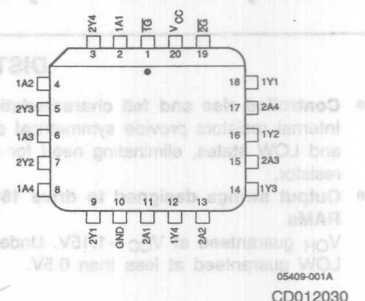
Am2965

Inputs		Outputs
\bar{G}	A	Y
H	X	Z
L	H	L
L	L	H

Am2966

Inputs		Outputs
\bar{G}	A	Y
H	X	Z
L	L	L
L	H	H

Plastic Leaded Chip Carrier

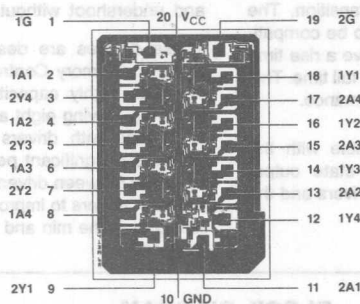


Note: Pin 1 is marked for orientation

*Also available in 20-Pin Small Outline package for Am2966 only; pinout identical to DIPs.

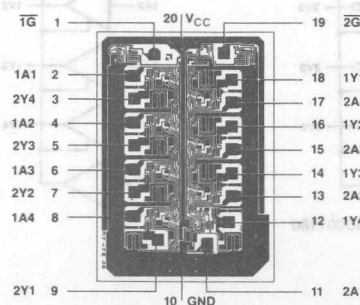
METALLIZATION AND PAD LAYOUT

Am2965



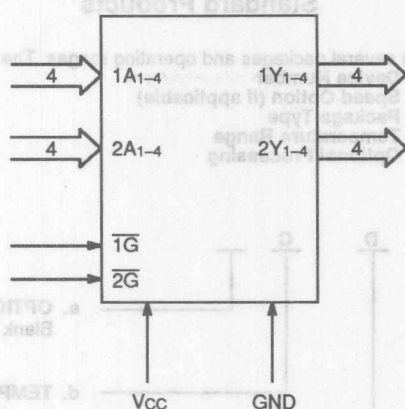
DIE SIZE 0.094" x 0.060"

Am2966



DIE SIZE 0.094" x 0.066"

LOGIC SYMBOL



Parameter	PD	PLCC	SOIC	Units
θ_{JA}	71	72	75	°C/Watt
θ_{JC}	22	18	16	

05409-002A

Valid Combinations	
Am2965	JC, PC
Am2966	JC, PC, SC

ORDERING INFORMATION

Standard Products

AMD standard products are available in several packages and operating ranges. The order number (Valid Combination) is formed by a combination of:

- Device Number
- Speed Option (If applicable)
- Package Type
- Temperature Range
- Optional Processing

AM2965
AM2966

D

C

e. OPTIONAL PROCESSING
Blank = Standard processing

d. TEMPERATURE RANGE
C = Commercial (0 to +70°C)

c. PACKAGE TYPE
P = 20-Pin Plastic DIP (PD 020)
J = 20-Pin Plastic Leaded Chip Carrier (PL 020)
S = 20-Pin Plastic Small Outline Package (SO 020)

b. SPEED OPTION
Not Applicable

a. DEVICE NUMBER/DESCRIPTION
Am2965/Am2966
Octal Dynamic Memory Drivers with Three-State Outputs

Valid Combinations

Valid Combinations	
AM2965	JC, PC
AM2966	JC, PC, SC

Valid Combinations

Valid Combinations list configurations planned to be supported in volume for this device. Consult the local AMD sales office to confirm availability of specific valid combinations or to check on newly released combinations, and to obtain additional data on AMD's standard military grade products.

MILITARY ORDERING INFORMATION

APL Products

AMD products for Aerospace and Defense applications are available in several packages and operating ranges. APL (Approved Products List) products are fully compliant with MIL-STD-883C requirements. The order number (Valid Combination) is formed by a combination of:

- Device Number
- Speed Option (if applicable)
- Package Type
- Temperature Range
- Optional Processing

AM2965
AM2966

/B

R

A

e. LEAD FINISH

A = Hot Solder Dip

d. PACKAGE TYPE

R = 20-Pin CERDIP (CD 020)

c. DEVICE CLASS

/B = Class B

b. SPEED OPTION

Not Applicable

a. DEVICE NUMBER/DESCRIPTION

Am2965/Am2966

Octal Dynamic Memory Drivers with Three-State Outputs

Valid Combinations

AM2965
AM2966

/BRA

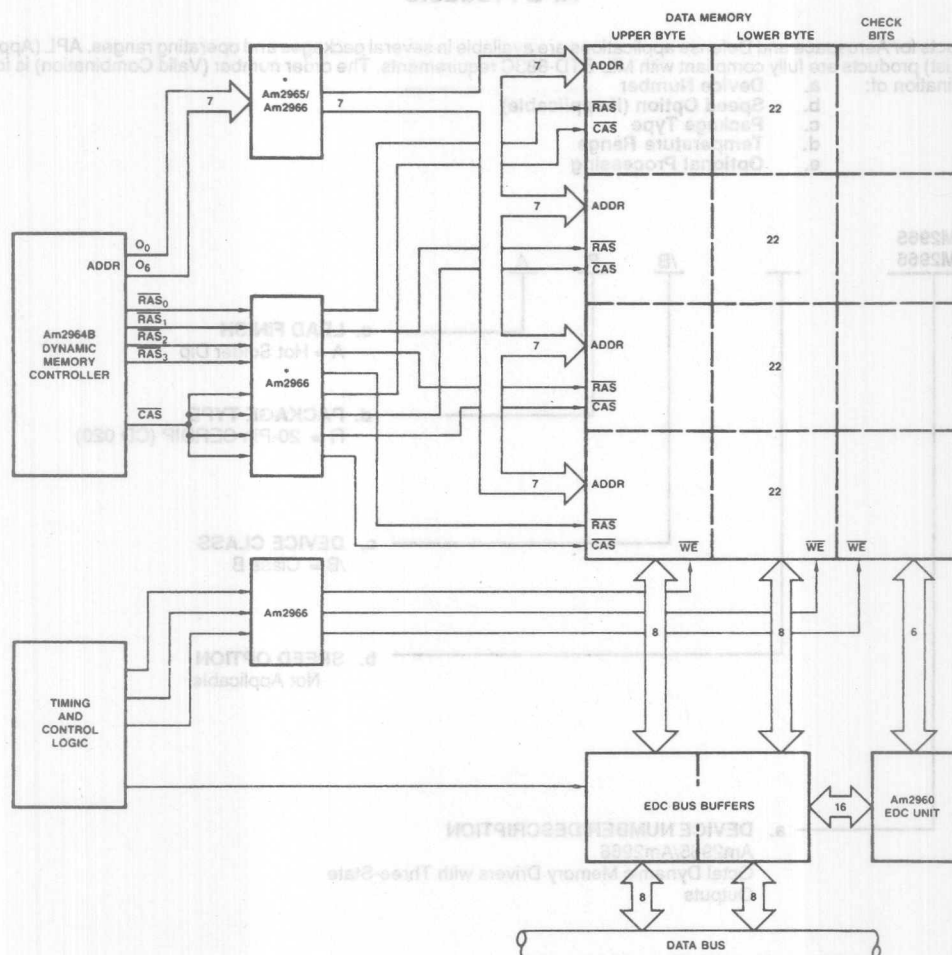
Valid Combinations

Valid Combinations list configurations planned to be supported in volume for this device. Consult the local AMD sales office to confirm availability of specific valid combinations, to check on newly released combinations.

Group A Tests

Group A tests consists of Subgroups
1, 2, 3, 7, 8, 9, 10, 11

APPLICATION



AF000401

*Address and RAS/CAS drivers each drive 22 RAM inputs at each output. Timing skew is minimized by using one device for address lines and one device for RAS/CAS, spreading the CAS loading over four drivers to equalize the capacitive load on each driver.

DYNAMIC MEMORY CONTROL WITH ERROR DETECTION AND CORRECTION

ABSOLUTE MAXIMUM RATINGS

Storage Temperature	-65°C to +150°C
Temperature (Case)	-55°C to +125°C
Under Bias	-55°C to +125°C
Supply Voltage to Ground Potential	
Continuous	-0.5V to +7.0V
DC Voltage Applied to Outputs For	
High Output State	-0.5V to V_{CC} Max
DC Input Voltage	-0.5V to +7.0V
DC Output Current, Into Outputs	200mA
DC Input Current	-30mA to +5.0mA

Stresses above those listed under ABSOLUTE MAXIMUM RATINGS may cause permanent device failure. Functionality at or above these limits is not implied. Exposure to absolute maximum ratings for extended periods may affect device reliability.

OPERATING RANGES

Commercial (C) Devices	
Temperature	0°C to +70°C
Supply Voltage	+4.75V to +5.25V
Military (M) Devices	
Temperature	-55°C to +125°C
Supply Voltage	+4.5V to +5.5V

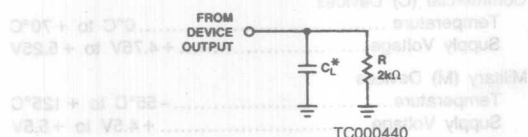
Operating ranges define those limits over which the functionality of the device is guaranteed.

DC CHARACTERISTICS over operating ranges unless otherwise specified (for APL Products, Group A Subgroups 1, 2, 3, 7, and 8 are tested unless otherwise noted)

Parameters	Descriptions	Test Conditions (Note 1)	Min	Typ (Note 2)	Max	Units
V_{OH}	Output HIGH Voltage	$V_{CC} = \text{MIN}$ $V_{IN} = V_{IH}$ or V_{IL}	$I_{OH} = -1\text{mA}$	$V_{CC} - 1.15$	$V_{CC} - 0.7V$	Volts
V_{OL}	Output LOW Voltage	$V_{CC} = \text{MIN}$ $V_{IN} = V_{IH}$ or V_{IL}	$I_{OL} = 1\text{mA}$ $I_{OL} = 12\text{mA}$		0.5 0.8	Volts
V_{IH}	Input HIGH Level	Guaranteed input logical HIGH voltage for all inputs	2.0			Volts
V_{IL}	Input LOW Level	Guaranteed input logical LOW voltage for all inputs			0.8	Volts
V_I	Input Clamp Voltage	$V_{CC} = \text{MIN}$, $I_{IN} = -18\text{mA}$			-1.2	Volts
I_{IL}	Input LOW Current	$V_{CC} = \text{MAX}$, $V_{IN} = 0.4V$	DATA 1G, 2G		-200 -400	μA
I_{IH}	Input HIGH Current	$V_{CC} = \text{MAX}$, $V_{IN} = 2.7V$			20	μA
I_I	Input HIGH Current	$V_{CC} = \text{MAX}$, $V_{IN} = 7.0V$			0.1	mA
I_{OZH}	Off-State Current	$V_O = 2.7V$			100	μA
I_{OZL}	Off-State Current	$V_O = 0.4V$			-200	μA
I_{OL}	Output Sink Current	$V_{OL} = 2.0V$	50			mA
I_{OH}	Output Source Current	$V_{OH} = 2.0V$	-35			mA
I_{SC}	Output Short Circuit Current (Note 3)	$V_{CC} = \text{MAX}$	-60 (see I_{OH})		-200	mA
I_{CC}	Supply Current	Am2965	All Outputs HIGH		24	mA
			All Outputs LOW	$V_{CC} = \text{MAX}$ Outputs Open	86	
			All Outputs Hi-Z		86	
		Am2966	All Outputs HIGH		53	
			All Outputs LOW	$V_{CC} = \text{MAX}$ Outputs Open	92	
			All Outputs Hi-Z		116	

- Notes: 1. For conditions shown as MIN or MAX, use the appropriate value specified under Operating Range for the applicable device type.
2. Typical limits are at $V_{CC} = 5.0V$, 25°C ambient and maximum loading.
3. Not more than one output should be shorted at a time. Duration of the short circuit test should not exceed one second.

SWITCHING TEST CIRCUIT



* t_{pd} specified at $C = 50$ and 500pF .
Figure 1. Capacitive Load Switching.

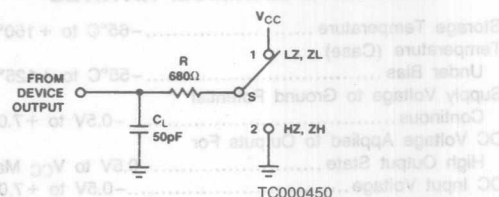
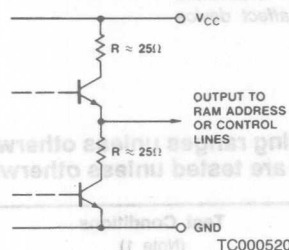


Figure 2. Three-State Enable/Disable.

TYPICAL OUTPUT DRIVER



SWITCHING CHARACTERISTICS ($T_A = +25^\circ\text{C}$, $V_{CC} = 5.0\text{ V}$) for APL Products, Group A Subgroups 9, 10, and 11 are tested unless otherwise noted)

Parameters	Description	Test Conditions	Min	Typ	Max	Units
t_{PLH}	Propagation Delay Time from LOW-to-HIGH Output	$C_L = 0\text{pF}$ Figure 1 Test Circuit Figure 3 Voltage Levels and Waveforms		6	(Note 4)	ns
		$C_L = 50\text{pF}$	6	9	15	
		$C_L = 500\text{pF}$	18	22	30	
t_{PHL}	Propagation Delay Time from HIGH-to-LOW Output	$C_L = 0\text{pF}$ $C_L = 50\text{pF}$ $C_L = 500\text{pF}$		4 7 22	(Note 4) 15 30	ns
t_{PLZ}	Output Disable Time from LOW, HIGH	Figures 2 and 4, $S = 1$		11	20	
t_{PHZ}	Output Enable Time from LOW, HIGH	Figures 2 and 4, $S = 2$		6.5	12	
t_{PZL}	Output Enable Time from LOW, HIGH	Figures 2 and 4, $S = 1$		12	20	ns
t_{PZH}	Output Enable Time from LOW, HIGH	Figures 2 and 4, $S = 2$		12	20	
t_{SKEW}	Output-to-Output Skew	Figures 1 and 3, $C_L = 50\text{pF}$		± 0.5	± 3.0 (Note 5)	ns
V_{ONP}	Output Voltage Undershoot	Figures 1 and 3, $C_L = 50\text{pF}$		0	-0.5	Volts

SWITCHING CHARACTERISTICS over operating ranges unless otherwise specified (Note 6)

Parameters	Description	Test Conditions	COMMERCIAL		MILITARY		Units
			Min	Max	Min	Max	
t_{PLH}	Propagation Delay Time LOW-to-HIGH Output	Figures 1 and 3 $C_L = 50\text{pF}$ $C_L = 500\text{pF}$	4 18	17 35	4 18	20 40	ns
t_{PHL}	Propagation Delay Time HIGH-to-LOW Output	$C_L = 50\text{pF}$ $C_L = 500\text{pF}$	4 18	17 35	4 18	20 40	
t_{PLZ}	Output Disable Time from LOW, HIGH	Figures 2 and 4 $S = 1$ $S = 2$		24 16		24 16	ns
t_{PZL}	Output Enable Time from LOW, HIGH	Figures 2 and 4 $S = 1$ $S = 2$		28 28		28 28	
V_{ONP}	Output Voltage Undershoot	Figures 1 and 3, $C_L = 50\text{pF}$		-0.5		-0.5	Volts

Notes: 4. Typical time shown for reference only - not tested.

5. Time Skew specification is guaranteed by design but not tested.

6. AC performance over the operating temperature range is guaranteed by testing defined in Group A, Subgroup 9.

7. $T_C = -55$ to $+125^\circ\text{C}$ for Flatpak versions.

TYPICAL SWITCHING CHARACTERISTICS

SWITCHING TEST WAVEFORMS

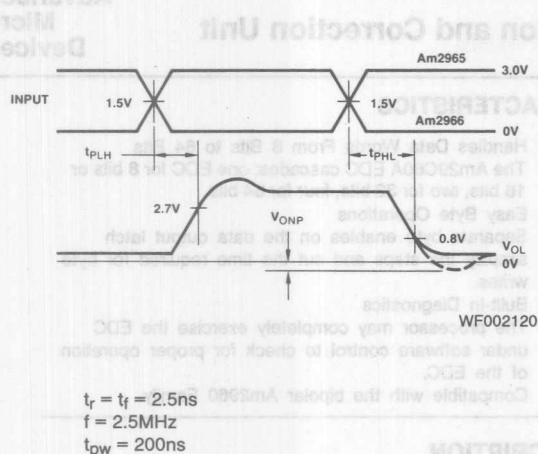


Figure 3. Output Drive Levels.

The RAM Driver symmetrical output design offers significant improvement over a standard Schottky output by providing a balanced drive output impedance ($\approx 25\Omega$ both HIGH and LOW), and by pulling up to MOS V_{OH} levels ($V_{CC} - 1.5V$). External resistors, not required with the RAM Driver, protect standard Schottky drivers from error causing undershoot but also slow the output rise by adding to the internal R.

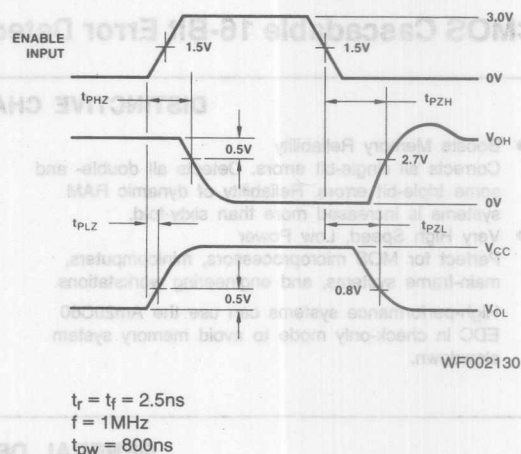


Figure 4. Three-State Control Levels.

The RAM Driver is optimized to drive LOW at maximum speed based on safe undershoot control and to drive HIGH with a symmetrical speed characteristic. This is an optimum approach, because the dominant RAM loading characteristic is input capacitance.

The curves shown below provide performance characteristics typical of both the inverting (Am2965) and non-inverting (Am2966) RAM Drivers.

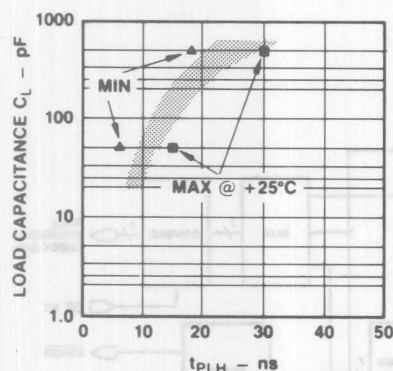


Figure 5. t_{PLH} for $V_{OH} = 2.7V$ vs. C_L .

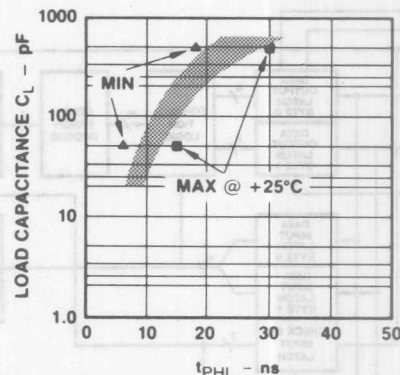


Figure 6. t_{PHL} for $V_{OH} = 0.8V$ vs. C_L .

The curves above depict the typical t_{PLH} and t_{PHL} for the RAM Driver outputs as a function of load capacitance. The minimums and maximums are shown for worst case design. The typical band is provided as a guide for intermediate capacitive loads.

Am29C60A

CMOS Cascadable 16-Bit Error Detection and Correction Unit

Advanced
Micro
Devices

DISTINCTIVE CHARACTERISTICS

- Boosts Memory Reliability
Corrects all single-bit errors. Detects all double- and some triple-bit errors. Reliability of dynamic RAM systems is increased more than sixty-fold.
- Very High Speed, Low Power
Perfect for MOS microprocessors, minicomputers, main-frame systems, and engineering workstations.
High-performance systems can use the Am29C60 EDC in check-only mode to avoid memory system slowdown.
- Handles Data Words From 8 Bits to 64 Bits
The Am29C60A EDC cascades: one EDC for 8 bits or 16 bits, two for 32 bits, four for 64 bits.
- Easy Byte Operations
Separate byte enables on the data output latch simplify the steps and cut the time required for byte writes.
- Built-In Diagnostics
The processor may completely exercise the EDC under software control to check for proper operation of the EDC.
- Compatible with the bipolar Am2960 Family.

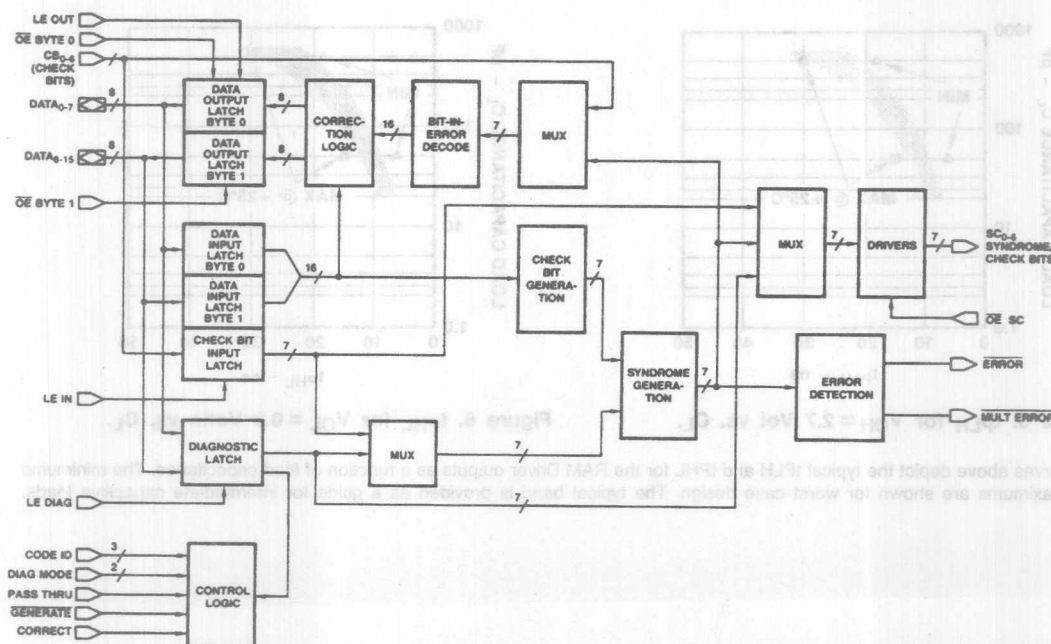
GENERAL DESCRIPTION

The Am29C60A Error Detection and Correction Unit (EDC) contains the logic necessary to generate check bits on a 16-bit data field according to a modified Hamming Code, and to correct the data word when check bits are supplied. Operating on data read from memory, the Am29C60A corrects any single-bit errors and detects all double- and some triple-bit errors. For 16-bit words, 6 check bits are used. The Am29C60A is expandable to operate on 32-bit words (7

check bits) and 64-bit words (8 check bits). In all configurations, the device makes the error syndrome available on separate outputs for data logging.

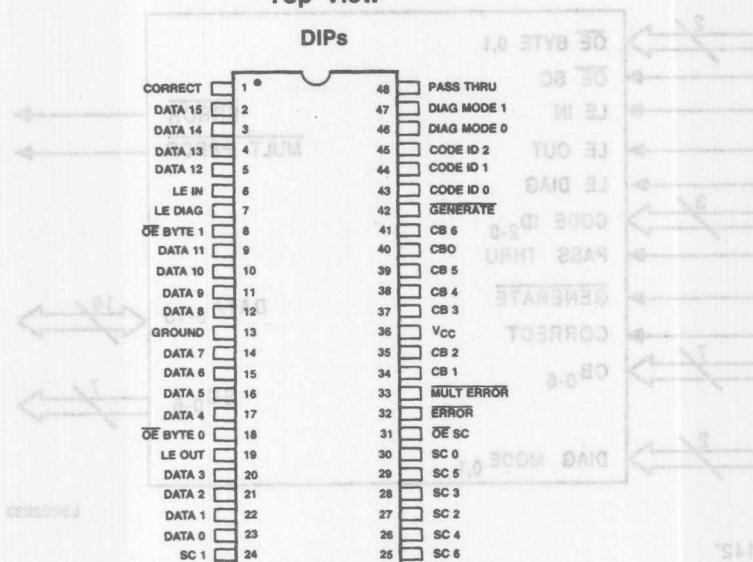
The Am29C60A also features two diagnostic modes in which diagnostic data can be forced into portions of the chip to simplify device testing and to execute system diagnostic functions.

BLOCK DIAGRAM



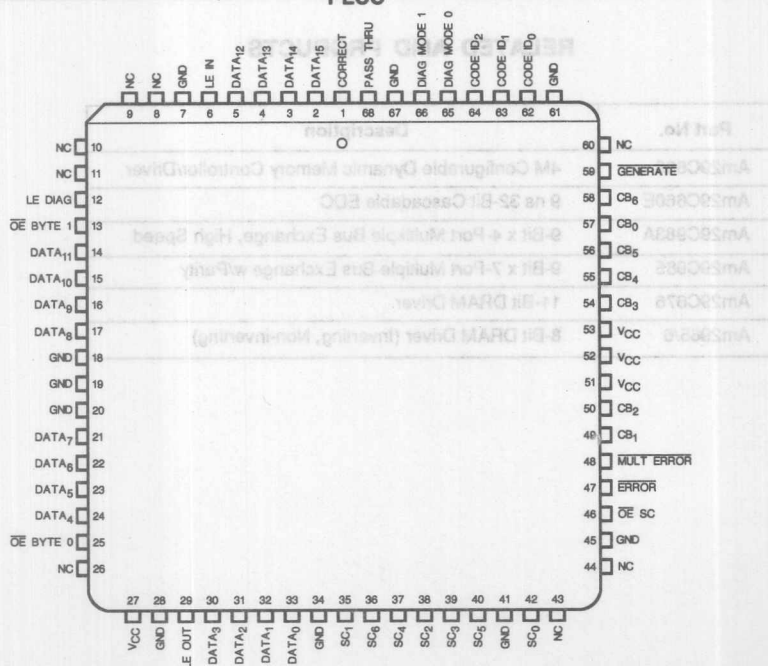
BD001261

CONNECTION DIAGRAMS Top View



CD001421

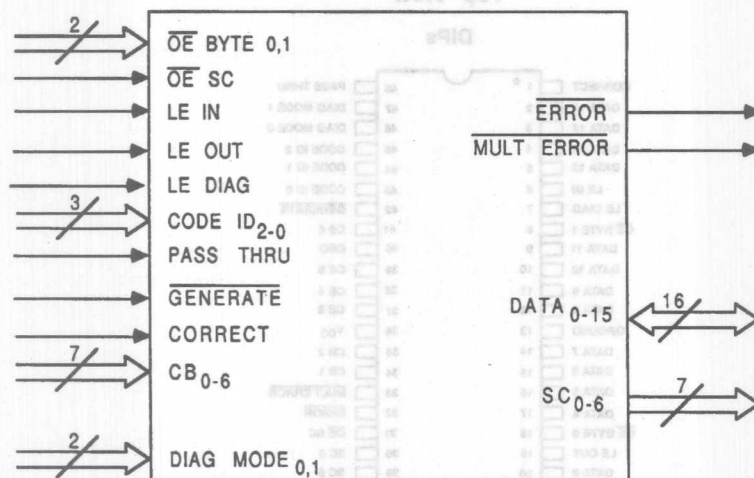
PLCC



CD010232

Note: Pin 1 is marked for orientation.

LOGIC SYMBOL



LS002833

Die Size: .137" x .142"
Gate Count: 875

RELATED AMD PRODUCTS

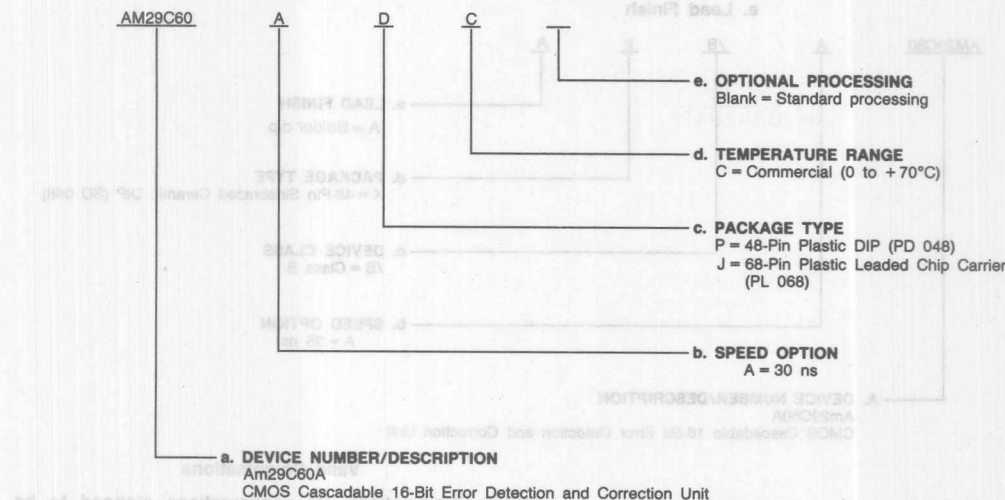
Part No.	Description
Am29C668	4M Configurable Dynamic Memory Controller/Driver
Am29C660E	9 ns 32-Bit Cascadable EDC
Am29C983A	9-Bit x 4-Port Multikple Bus Exchange, High Speed
Am29C985	9-Bit x 7-Port Multiple Bus Exchange w/Parity
Am29C676	11-Bit DRAM Driver
Am2965/6	8-Bit DRAM Driver (Inverting, Non-inverting)

ORDERING INFORMATION

Standard Products

AMD standard products are available in several packages and operating ranges. The order number (Valid Combination) is formed by a combination of:

- a. Device Number
- b. Speed Option (if applicable)
- c. Package Type
- d. Temperature Range
- e. Optional Processing



Valid Combinations	
AM29C60A	PC, JC

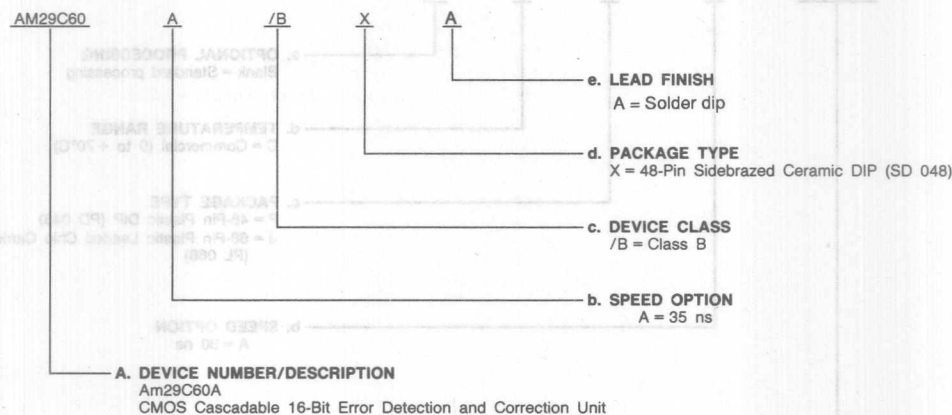
Valid Combinations list configurations planned to be supported in volume for this device. Consult the local AMD sales office to confirm availability of specific valid combinations, to check on newly released combinations, and to obtain additional data on AMD's standard military grade products.

MILITARY ORDERING INFORMATION

APL Products

AMD products for Aerospace and Defense applications are available in several packages and operating ranges. APL (Approved Products List) products are fully compliant with MIL-STD-883C requirements. The order number (Valid Combination) for APL products is formed by a combination of:

- a. Device Number
- b. Speed Option (if applicable)
- c. Device Class
- d. Package Type
- e. Lead Finish



Valid Combinations	
AM29C60A	/BXA

Valid Combinations

Valid Combinations list configurations planned to be supported in volume for this device. Consult the local AMD sales office to confirm availability of specific valid combinations or to check for newly released valid combinations.

Group A Tests

Group A tests consists of Subgroups
1, 2, 3, 7, 8, 9, 10, 11

PIN DESCRIPTION

CB₀₋₆ Check Bits (Input)

The check bit lines are used to input check bits for error detection. Also used to input syndrome bits for error correction in 32- and 64-bit configurations.

CODE ID₂₋₀ Code Identification (Input)

These three bits identify the size of the total data word to be processed and which 16-bit slice of larger data words a particular EDC is processing. The three allowable data word sizes are 16, 32, and 64 bits, and their respective modified Hamming codes are designated 16/22, 32/39, and 64/72. Special CODE ID input 001 (ID₂, ID₁, ID₀) is also used to instruct the EDC to take the signals CODE ID₂₋₀, DIAG MODE₀₋₁, CORRECT, and PASS THRU from the Diagnostic Latch, rather than from the input control lines.

CORRECT Correct (Input)

When HIGH, this signal allows the correction network to correct any single-bit error in the Data Input Latch (by complementing the bit-in-error) before putting it into the Data Output Latch. When the signal is LOW, the EDC drives data directly from the Data Input Latch to the Data Output Latch without correction.

DATA₀₋₁₅ Data (Input/Output; Three State)

These bidirectional data lines provide input to the Data Input Latch and Diagnostic Latch, and receive output from the Data Output Latch. DATA₀ is the least significant bit; DATA₁₅ the most significant.

DIAG MODE₀₋₁ Diagnostic Mode Select (Input)

These two lines control the initialization and diagnostic operation of the EDC.

ERROR Error Detected (Output)

When the EDC is in Detect or Correct Mode, this output goes LOW if one or more syndrome bits are asserted, indicating one or more bit errors in the data or check bits. If no syndrome bits are asserted, there are no errors detected and the output will be HIGH. In Generate Mode, ERROR is forced HIGH. In a 64-bit configuration, ERROR must be externally implemented.

GENERATE Generate Check Bits (Input)

When this input is LOW, the EDC is in the Check Bit Generate Mode. When HIGH, the EDC is in the Detect Mode or Correct Mode.

In the Generate Mode, the circuit generates the check bits or partial check bits specific to the data in the Data Input Latch. The generated check bits are placed on the SC outputs.

In the Detect or Correct Modes, the EDC detects single and multiple errors and generates syndrome bits based on the contents of the Data Input Latch and Check Bit Input Latch. In Correct Mode, single-bit errors are also automatically corrected; corrected data is placed at the inputs of the Data Output Latch. The syndrome result is placed on the SC outputs, and indicates in a coded form the number of errors and the bit-in-error.

LE DIAG Diagnostic Latch Enable (Input)

When this input is HIGH, the Diagnostic Latch follows the 16-bit data on the input lines. When it is LOW, the outputs of the Diagnostic Latch are latched to their previous states. The Diagnostic Latch holds diagnostic check bits and internal control signals for CODE ID₂₋₀, DIAG MODE₀₋₁, CORRECT, and PASS THRU.

LE IN Latch Enable - Data Input Latch (Input)

This input controls latching of the input data. When HIGH, the Data Input Latch and Check Bit Input Latch follow the input data and input check bits. When LOW, the Data Input Latch and Check Bit Input Latch are latched to their previous state.

LE OUT Latch Enable - Data Output Latch (Input)

This input controls the latching of the Data Output Latch. When it is LOW, the Data Output Latch is latched to its previous state. When it is HIGH, the Data Output Latch follows the output of the Data Input Latch as modified by the correction logic network. In Correct Mode, single-bit errors are corrected by the network before loading into the Data Output Latch. In Detect Mode, the contents of the Data Input Latch are passed unchanged through the correction network into the Data Output Latch. The inputs to the Data Output Latch are unspecified if the EDC is in Generate Mode.

MULT ERROR Multiple Errors Detected (Output)

When the EDC is in Detect or Correct Mode, this output, if LOW, indicates that two or more bit errors have been detected. If HIGH, either one or no errors have been detected. In Generate Mode, MULT ERROR is forced HIGH. In a 64-bit configuration, MULT ERROR must be externally implemented.

OE BYTE 0, 1 Output Enable Bytes 0, 1 (Input)

These lines control the three-state outputs for each of the two bytes of the Data Output Latch. When LOW, these lines enable the Data Output Latch, and when HIGH, these lines force the Data Output Latch into the high-impedance state. The two enable lines can be separately activated to enable only one byte of the Data Output Latch at a time.

OE SC Output Enable, Syndrome/Check Bits (Input)

When this input is LOW, the three-state output lines SC₀₋₆ are enabled. When the input is HIGH, the SC outputs are in the high-impedance state.

PASS THRU Pass Thru (Input)

This line, when HIGH, forces the contents of the Check Bit Input Latch onto the Syndrome/Check Bit outputs (SC₀₋₆) and the unmodified contents of the Data Input Latch onto the inputs of the Data Output Latch.

SC₀₋₆ Syndrome/Check Bits (Output; Three State)

These seven lines hold the check/partial-check bits when the EDC is in Generate Mode, and hold the syndrome/partial syndrome bits when the device is in Detect or Correct Modes. These are three-state outputs.

FUNCTIONAL DESCRIPTION

The CMOS version saves system power with no loss in performance. The Am29C60A is a performance upgrade of the Am29C60 but is not a parametric equivalent to the bipolar Am2960A.

The CMOS EDC circuit contains proprietary output buffers to decrease the on-chip-generated noise caused by current changes

through fast logic. This minimizes "ground bounce" and ensures proper chip performance of these high-speed CMOS solutions in the system environment.

Please refer to EDC Product Specifications Booklet (Literature #03565E/0) for detailed functional description and applications information.

ABSOLUTE MAXIMUM RATINGS

Storage Temperature	-65°C to +150°C
Temperature (Case)	
Under Bias	-55°C to +125°C
Supply Voltage to Ground Potential	
Continuous	-0.5 V to +7.0 V
DC Voltage Applied to Outputs For	
High Output State	-0.5 V to V _{CC} Max.
DC Input Voltage	-0.5 V to +5.5 V
DC Input Current	-30 mA to +5.0 mA

Stresses above those listed under ABSOLUTE MAXIMUM RATINGS may cause permanent device failure. Functionality at or above these limits is not implied. Exposure to absolute maximum ratings for extended periods may affect device reliability.

OPERATING RANGES

Commercial (C) Devices	
Ambient Temperature (T _A)	0°C to +70°C
Supply Voltage	+4.75 V to +5.25 V
Military (M) Devices	
Case Temperature (T _C)	-55°C to +125°C
Supply Voltage	+4.5 V to +5.5 V

Operating ranges define those limits between which the functionality of the device is guaranteed.

DC CHARACTERISTICS over operating range unless otherwise specified (for APL Products, Group A, Subgroups 1, 2, 3 are tested unless otherwise noted)

Parameter Symbol	Parameter Description	Test Conditions (Note 1)	Min.	Max.	Unit
V _{OH}	Output HIGH Voltage	V _{CC} = Min., V _{IN} = V _{IH} or V _{IL} I _{OH} = -300 μ A MIL I _{OH} = -12 mA COM'L I _{OH} = -15 mA	V _{CC} - 0.2 2.4 2.4		V
V _{OL}	Output LOW Voltage	V _{CC} = Min., V _{IN} = V _{IH} or V _{IL} I _{OL} = 300 μ A MIL I _{OL} = 8 mA COM'L I _{OL} = 16 mA		0.2 0.5 0.5	V
V _{IH}	Input HIGH Voltage	Guaranteed Input Logical HIGH Voltage for all Inputs (Note 5)	2.0		V
V _{IL}	Input LOW Voltage	Guaranteed Input Logical LOW Voltage for all Inputs (Note 5)		0.8	V
I _{IL}	Input LOW Current	V _{CC} = Max., V _{IN} = 0.5 V DATA ₀₋₁₅ (Note 3) All Other Inputs		-10 -10	μ A
I _{IH}	Input HIGH Current	V _{CC} = Max., V _{IN} = 2.7 V DATA ₀₋₁₅ (Note 3) All Other Inputs		10 10	μ A
I _I	Input HIGH Current	V _{CC} = Max.,		10	μ A
I _{OZH} I _{OZL}	Off State (High-Impedance) Output Current	V _{CC} = Max., DATA ₀₋₁₅ V _O = 2.4 V _O = 0.5 SC ₀₋₆ V _O = 2.4 V _O = 0.5		40 -40 40 -40	μ A
I _{OS}	Output Short-Circuit Current (Note 2)	V _{CC} = Max. V _O = 0 V	-30	-140	mA
I _{CC}	Power Supply (Note 4)	V _{CC} = Max., COM'L MIL V _{CC} = 5.0 V (Note 6) T _A = +25°C		50 50 30	mA
I _{CCOC}	Quiescent Power Supply (CMOS)	V _{CC} = Max., COM'L MIL V _{CC} = 5.0 V (Note 6) T _A = +25°C		5 5 2	mA

- Notes: 1. For conditions shown as Min. or Max., use the appropriate value specified under Operating Range for the applicable device type.
2. Not more than one output should be shorted at a time. Duration of the short-circuit test should not exceed one second.
3. These are three-state outputs internally connected to TTL inputs. Input characteristics are measured with outputs disabled (high impedance).
4. Worst-case I_{CC} is at minimum temperature. Test conditions: C_I = 50 pF, f = 10 MHz, V_{IN} = 50% duty cycle for all inputs at 3.4 V and 0.4 V, OE = GND.
5. These input levels provide zero noise immunity and should only be tested in a static, noise-free environment.
6. Not production tested. Typical I_{CC} (V_{CC} = 5.0 V and T_A = 25°C) represents nominal units.

SWITCHING CHARACTERISTICS over COMMERCIAL operating range (Notes 1 and 2)

The following table specifies the guaranteed device performance over the commercial operating range of 0°C to +70°C (ambient), with V_{CC} 4.75 to 5.25 V. All input switching is between 0 V and 3 V at 1 V/ns, and measurements are made at 1.5 V. All outputs have maximum DC load. All units are in nanoseconds (ns).

No.	Parameter Symbol	Data Path Description		Am29C60A	
		From Input	To Output	Min.	Max.
1	t_{PD}	DATA ₀₋₁₅ (Note 3)	SC ₀₋₆		20
			DATA ₀₋₁₅		30
			ERROR		20
			MULT ERROR		23
2	t_{PD}	CB ₀₋₆ (CODE ID ₂₋₀ 000, 011)	SC ₀₋₆ (Note 7)		14
			DATA ₀₋₁₅		25
			ERROR		20
			MULT ERROR		23
3	t_{PD}	CB ₀₋₆ (CODE ID ₂₋₀ 010, 100, 101, 110, 111)	SC ₀₋₆ (Note 7)		17
			DATA ₀₋₁₅		25
			ERROR		20
			MULT ERROR		23
4	t_{PD}	GENERATE	SC ₀₋₆ (Note 7)		17
			DATA ₀₋₁₅		25
			ERROR (Note 7)		16
			MULT ERROR		17
5	t_{PD}	CORRECT (Not Internal Control Mode)	SC ₀₋₆		-
			DATA ₀₋₁₅		20
			ERROR		-
			MULT ERROR		-
6	t_{PD}	DIAG MODE (Not Internal Control Mode)	SC ₀₋₆ (Note 7)		22
			DATA ₀₋₁₅		27
			ERROR (Note 7)		19
			MULT ERROR (Note 7)		21
7	t_{PD}	PASS THRU (Not Internal Control Mode)	SC ₀₋₆		22
			DATA ₀₋₁₅		25
			ERROR		18
			MULT ERROR		21
8	t_{PD}	CODE ID ₂₋₀	SC ₀₋₆		23
			DATA ₀₋₁₅		28
			ERROR		25
			MULT ERROR		28

Notes: See notes at end of this section.

SWITCHING CHARACTERISTICS over **COMMERCIAL** operating range (Cont'd.)

No.	Parameter Symbol	Data Path Description		Am29C60A	
		From Input	To Output	Min.	Max.
9	t _{PD}	LE IN (From latched to transparent)	SC ₀₋₆		22
			DATA ₀₋₁₅		32
			ERROR		22
			MULT ERROR		25
10	t _{PD}	LE OUT (From latched to transparent)	SC ₀₋₆		-
			DATA ₀₋₁₅		13
			ERROR		-
			MULT ERROR		-
11	t _{PD}	LE DIAG (From latched to transparent; Not Internal Control Mode)	SC ₀₋₆		22
			DATA ₀₋₁₅		32
			ERROR		22
			MULT ERROR		25
12	t _{PD}	Internal Control Mode: LE DIAG (From latched to transparent)	SC ₀₋₆		28
			DATA ₀₋₁₅		38
			ERROR		28
			MULT ERROR		31
13	t _{PD}	Internal Control Mode: DATA ₀₋₁₅ (Via Diagnostic latch)	SC ₀₋₆		28
			DATA ₀₋₁₅		38
			ERROR		28
			MULT ERROR		31
14	t _{SET}	DATA ₀₋₁₅	LE IN	5	
15	t _{HOLD}	(Notes 4, 5)		3	
16	t _{SET}	CB ₀₋₆		5	
17	t _{HOLD}	(Notes 4, 5)		3	
18	t _{SET}	DATA ₀₋₁₅	LE OUT	24	
19	t _{HOLD}	(Notes 4, 5)		2	
20	t _{SET}	CB ₀₋₆		21	
21	t _{HOLD}	(CODE ID 000, 011)		0	
22	t _{SET}	CB ₀₋₆		21	
23	t _{HOLD}	(CODE ID 010, 100, 101, 110, 111)		0	
24	t _{SET}	GENERATE		26	
25	t _{HOLD}	(Notes 4, 5)		0	
26	t _{SET}	CORRECT		22	
27	t _{HOLD}	(Notes 4, 5)		0	
28	t _{SET}	DIAG MODE		22	
29	t _{HOLD}	(Notes 4, 5)		0	
30	t _{SET}	PASS THRU		22	
31	t _{HOLD}	(Notes 4, 5)		0	
32	t _{SET}	CODE ID ₂₋₀		25	
33	t _{HOLD}	(Notes 4, 5)		0	

Notes: See notes at end of this section.

SWITCHING CHARACTERISTICS over **COMMERCIAL** operating range (Cont'd.)

No.	Parameter Symbol	Data Path Description		Am29C60A	
		From Input	To Output	Min.	Max.
34	tSET	LE IN (Notes 4, 5)	LE OUT	28	
35	tHOLD			0	
36	tSET	DATA ₀ – 15 (Notes 4, 5)	LE DIAG	3	
37	tHOLD			5	
38	tEN	OE BYTE 0,1 ENABLE (Note 6)	DATA ₀ – 15		14
39	tDIS				23
40	tEN	OE SC DISABLE (Note 6)	SC ₀ – 6		16
41	tDIS				21
42	tpw	MINIMUM PULSE WIDTH: LE IN, LE OUT, LE DIAG		12	

Notes: 1. C_L = 50 pF.

2. Certain parameters are combinational propagation delay calculations.

3. Data IN or LE IN to Correct Data Out measurement requires timing as shown in the Switching Waveforms.

4. Setup and Hold times relative to Latch Enables (Latching up data).

5. Setup and Hold times are not tested, but are guaranteed by characterization.

6. Output disable tests specified with C_L = 5 pF and measured to 0.5 V change of output voltage level. Testing is performed at C_L = 50 pF and correlated to C_L = 5 pF.

7. Not production tested. Guaranteed by characterization.

SWITCHING CHARACTERISTICS over **MILITARY** operating range (for APL Products, Group A, Subgroups 9, 10, 11 are tested unless otherwise noted) (Notes 1 and 2)

The following table specifies the guaranteed device performance over the Military operating range of -55°C to $+125^{\circ}\text{C}$ (case), with V_{CC} 4.5 to 5.5 V. All input switching is between 0 V and 3 V at 1 V/ns and measurements are made at 1.5 V. All outputs have maximum DC load. All units are in nanoseconds (ns).

No.	Parameter Symbol	Data Path Description		Am29C60A	
		From Input	To Output	Min.	Max.
1	t_{PD}	DATA ₀₋₁₅ (Note 3)	SC ₀₋₆		24
			DATA ₀₋₁₅		35
			ERROR		24
			MULT ERROR		27
2	t_{PD}	CB ₀₋₆ (CODE ID ₂₋₀ 000, 011)	SC ₀₋₆ (Note 7)		17
			DATA ₀₋₁₅		28
			ERROR		24
			MULT ERROR		27
3	t_{PD}	CB ₀₋₆ (CODE ID ₂₋₀ 010, 100, 101, 110, 111)	SC ₀₋₆ (Note 7)		19
			DATA ₀₋₁₅		28
			ERROR		24
			MULT ERROR		27
4	t_{PD}	GENERATE	SC ₀₋₆ (Note 7)		20
			DATA ₀₋₁₅		28
			ERROR (Note 7)		21
			MULT ERROR		25
5	t_{PD}	CORRECT (Not Internal Control Mode)	SC ₀₋₆		-
			DATA ₀₋₁₅		25
			ERROR		-
			MULT ERROR		-
6	t_{PD}	DIAG MODE (Not Internal Control Mode)	SC ₀₋₆ (Note 7)		25
			DATA ₀₋₁₅		28
			ERROR (Note 7)		21
			MULT ERROR (Note 7)		24
7	t_{PD}	PASS THRU (Not Internal Control Mode)	SC ₀₋₆		25
			DATA ₀₋₁₅		28
			ERROR		21
			MULT ERROR		24
8	t_{PD}	CODE ID ₂₋₀	SC ₀₋₆		26
			DATA ₀₋₁₅		31
			ERROR		28
			MULT ERROR		31

Notes: See notes at end of this section.

SWITCHING CHARACTERISTICS over MILITARY operating range (Cont'd.)

		Data Path Description		Am29C60A	
No.	Parameter Symbol	From Input	To Output	Min.	Max.
9	t _{PD}	LE IN (From latched to transparent)	SC ₀₋₆		26
			DATA ₀₋₁₅		37
			ERROR		26
			MULT ERROR		29
10	t _{PD}	LE OUT (From latched to transparent)	SC ₀₋₆		-
			DATA ₀₋₁₅		16
			ERROR		-
			MULT ERROR		-
11	t _{PD}	LE DIAG (From latched to transparent; Not Internal Control Mode)	SC ₀₋₆		24
			DATA ₀₋₁₅		37
			ERROR		26
			MULT ERROR		29
12	t _{PD}	Internal Control Mode: LE DIAG (From latched to transparent)	SC ₀₋₆		30
			DATA ₀₋₁₅		43
			ERROR		32
			MULT ERROR		35
13	t _{PD}	Internal Control Mode: DATA ₀₋₁₅ (Via Diagnostic latch)	SC ₀₋₆		30
			DATA ₀₋₁₅		43
			ERROR		32
			MULT ERROR		35
14	t _{SET}	DATA ₀₋₁₅ (Notes 4, 5)	LE IN	5	
†15	t _{HOLD}			3	
16	t _{SET}	CB ₀₋₆ (Notes 4, 5)	LE IN	5	
†17	t _{HOLD}			3	
18	t _{SET}	DATA ₀₋₁₅ (Notes 4, 5)	LE OUT	27	
†19	t _{HOLD}			2	
20	t _{SET}	CB ₀₋₆ (CODE ID 000, 011)		24	
†21	t _{HOLD}			0	
22	t _{SET}	CB ₀₋₆ (Notes 4, 5) (CODE ID 010, 100, 101, 110, 111)		24	
†23	t _{HOLD}			0	
24	t _{SET}	GENERATE (Notes 4, 5)		29	
†25	t _{HOLD}			0	
26	t _{SET}	CORRECT (Notes 4, 5)		25	
†27	t _{HOLD}			0	
28	t _{SET}	DIAG MODE (Notes 4, 5)		25	
†29	t _{HOLD}			0	
30	t _{SET}	PASS THRU (Notes 4, 5)		25	
†31	t _{HOLD}			0	
32	t _{SET}	CODE ID ₂₋₀ (Notes 4, 5)		28	
†33	t _{HOLD}			0	

Notes: See notes at end of this section.

SWITCHING CHARACTERISTICS over MILITARY operating range (Cont'd.)

No.	Parameter Symbol	Data Path Description		Am29C60A	
		From Input	To Output	Min.	Max.
34	tSET	LE IN	(Notes 4, 5) LE OUT	30	
†35	tHOLD			0	
36	tSET	DATA0 – 15	(Notes 4, 5) LE DIAG	5	
†37	tHOLD			3	
38	tEN	OE BYTE 0,1 ENABLE	(Note 6) DATA0 – 15		28
39	tDIS				25
40	tEN	OE SC DISABLE	(Note 6) SC0 – 6		28
41	tDIS				25
42	tpw	MINIMUM PULSE WIDTH: LE IN, LE OUT, LE DIAG		12	

Notes: 1. C_L = 50 pF.

2. Certain parameters are combinational propagation delay calculations.

3. Data IN or LE IN to Correct Data Out measurement requires timing as shown in the Switching Waveforms.

4. Setup and Hold times relative to Latch Enables (Latching up data).

5. Setup and Hold times are not tested, but are guaranteed by characterization.

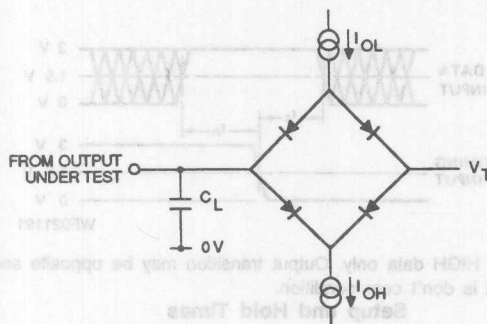
6. Output disable tests specified with C_L = 5 pF and measured to 0.5 V change of output voltage level. Testing is performed at C_L = 50 pF and correlated to C_L = 5 pF.

7. Not production tested. Guaranteed by characterization.

† = Not Included in Group A Tests.

13	10	Internal Control Mode: DATA-15 (Via Diagnostic Input)	MULTI ERROR	ERROR	DATA-15	20-0	MULTI ERROR	15	14	10
14	10									10
15	10									10
16	10									10
17	10	10	10	10	10	10	10	10	10	10
18	10	10	10	10	10	10	10	10	10	10
19	10	10	10	10	10	10	10	10	10	10
20	10	10	10	10	10	10	10	10	10	10
21	10	10	10	10	10	10	10	10	10	10
22	10	10	10	10	10	10	10	10	10	10
23	10	10	10	10	10	10	10	10	10	10
24	10	10	10	10	10	10	10	10	10	10
25	10	10	10	10	10	10	10	10	10	10
26	10	10	10	10	10	10	10	10	10	10
27	10	10	10	10	10	10	10	10	10	10
28	10	10	10	10	10	10	10	10	10	10
29	10	10	10	10	10	10	10	10	10	10
30	10	10	10	10	10	10	10	10	10	10
31	10	10	10	10	10	10	10	10	10	10
32	10	10	10	10	10	10	10	10	10	10
33	10	10	10	10	10	10	10	10	10	10
34	10	10	10	10	10	10	10	10	10	10
35	10	10	10	10	10	10	10	10	10	10
36	10	10	10	10	10	10	10	10	10	10
37	10	10	10	10	10	10	10	10	10	10
38	10	10	10	10	10	10	10	10	10	10
39	10	10	10	10	10	10	10	10	10	10
40	10	10	10	10	10	10	10	10	10	10
41	10	10	10	10	10	10	10	10	10	10
42	10	10	10	10	10	10	10	10	10	10
43	10	10	10	10	10	10	10	10	10	10
44	10	10	10	10	10	10	10	10	10	10
45	10	10	10	10	10	10	10	10	10	10
46	10	10	10	10	10	10	10	10	10	10
47	10	10	10	10	10	10	10	10	10	10
48	10	10	10	10	10	10	10	10	10	10
49	10	10	10	10	10	10	10	10	10	10
50	10	10	10	10	10	10	10	10	10	10
51	10	10	10	10	10	10	10	10	10	10
52	10	10	10	10	10	10	10	10	10	10
53	10	10	10	10	10	10	10	10	10	10
54	10	10	10	10	10	10	10	10	10	10
55	10	10	10	10	10	10	10	10	10	10
56	10	10	10	10	10	10	10	10	10	10
57	10	10	10	10	10	10	10	10	10	10
58	10	10	10	10	10	10	10	10	10	10
59	10	10	10	10	10	10	10	10	10	10
60	10	10	10	10	10	10	10	10	10	10
61	10	10	10	10	10	10	10	10	10	10
62	10	10	10	10	10	10	10	10	10	10
63	10	10	10	10	10	10	10	10	10	10
64	10	10	10	10	10	10	10	10	10	10
65	10	10	10	10	10	10	10	10	10	10
66	10	10	10	10	10	10	10	10	10	10
67	10	10	10	10	10	10	10	10	10	10
68	10	10	10	10	10	10	10	10	10	10
69	10	10	10	10	10	10	10	10	10	10
70	10	10	10	10	10	10	10	10	10	10
71	10	10	10	10	10	10	10	10	10	10
72	10	10	10	10	10	10	10	10	10	10
73	10	10	10	10	10	10	10	10	10	10
74	10	10	10	10	10	10	10	10	10	10
75	10	10	10	10	10	10	10	10	10	10
76	10	10	10	10	10	10	10	10	10	10
77	10	10	10	10	10	10	10	10	10	10
78	10	10	10	10	10	10	10	10	10	10
79	10	10	10	10	10	10	10	10	10	10
80	10	10	10	10	10	10	10	10	10	10
81	10	10	10	10	10	10	10	10	10	10
82	10	10	10	10	10	10	10	10	10	10
83	10	10	10	10	10	10	10	10	10	10
84	10	10	10	10	10	10	10	10	10	10
85	10	10	10	10	10	10	10	10	10	10
86	10	10	10	10	10	10	10	10	10	10
87	10	10	10	10	10	10	10	10	10	10
88	10	10	10	10	10	10	10	10	10	10
89	10	10	10	10	10	10	10	10	10	10
90	10	10	10	10	10	10	10	10	10	10
91	10	10	10	10	10	10	10	10	10	10
92	10	10	10	10	10	10	10	10	10	10
93	10	10	10	10	10	10	10	10	10	10
94	10	10	10	10	10	10	10	10	10	10
95	10	10	10	10	10	10	10	10	10	10
96	10	10	10	10	10	10	10	10	10	10
97	10	10	10	10	10	10	10	10	10	10
98	10	10	10	10	10	10	10	10	10	10
99	10	10	10	10	10	10	10	10	10	10
100	10	10	10	10	10	10	10	10	10	10

SWITCHING TEST CIRCUIT



AF004810

- Notes:
1. $C_L = 50$ pF for all tests except output enable/disable (includes scope probe, wiring, and stray capacitance without device in test fixture).
 2. $C_L = 5$ pF for output enable/disable tests.
 3. $V_T = 1.5$ V.

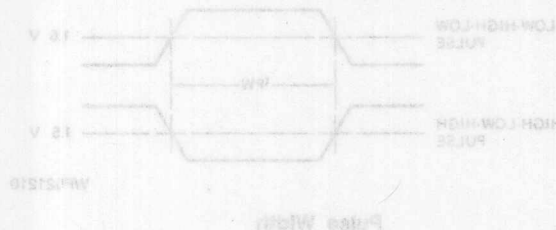
Notes on Testing

Incoming test procedures on this device should be carefully planned, taking into account the complexity and power levels of the part. The following notes may be useful.

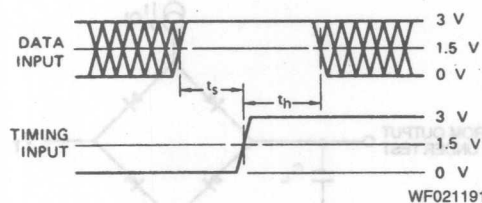
1. Ensure the part is adequately decoupled at the test head. Large changes in V_{CC} current as the device switches may cause erroneous function failures due to V_{CC} changes.
2. Do not leave inputs floating during any tests, as they may start to oscillate at high frequency.
3. Do not attempt to perform threshold tests at high speed. Following an input transition, ground current may change by as much as 400 mA in 5-8 ns. Inductance in the ground cable may allow the ground pin at the device to rise by hundreds of millivolts momentarily.

4. Use extreme care in defining input levels for AC tests. Many inputs may be changed at once, so there will be significant noise at the device pins and they may not actually reach V_{IL} or V_{IH} until the noise has settled. AMD recommends using $V_{IL} \leq 0$ V and $V_{IH} \geq 3$ V for AC tests.
5. To simplify failure analysis, programs should be designed to perform DC, Function, and AC tests as three distinct groups of tests.
6. Changing the CODE ID inputs can cause loss of data in some of the Am29C60 internal latches. Specifically, the entire checkbit latch and bits 6 and 7 of the diagnostic latch are indeterminate after a change in CODE ID inputs.

Logic simulations should store "x" (i.e., "don't care") in these bits after CODE ID change. Test programs should reload these registers before they are used.



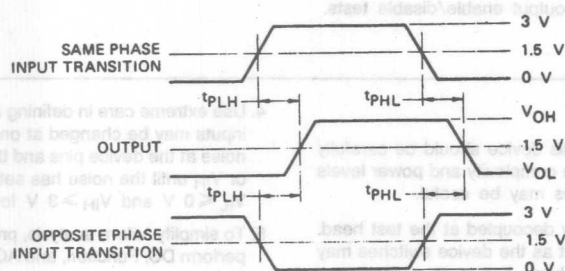
SWITCHING TEST WAVEFORMS



WF021191

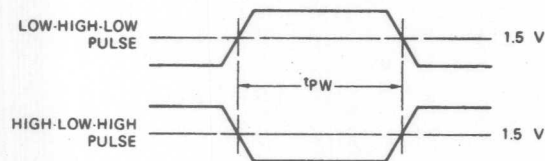
- Notes: 1. Diagram shown for HIGH data only. Output transition may be opposite sense.
2. Cross-hatched area is don't care condition.

Setup and Hold Times



WF021200

Propagation Delay



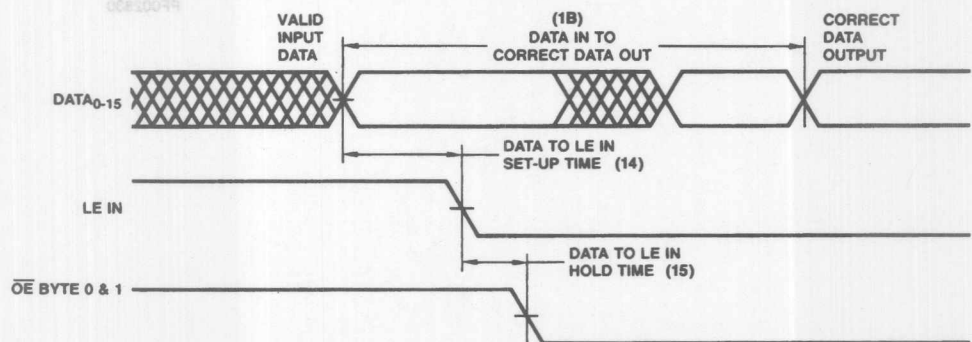
WF021210

Pulse Width

SWITCHING WAVEFORMS KEY TO SWITCHING WAVEFORMS

WAVEFORM	INPUTS	OUTPUTS
	MUST BE STEADY	WILL BE STEADY
	MAY CHANGE FROM H TO L	WILL BE CHANGING FROM H TO L
	MAY CHANGE FROM L TO H	WILL BE CHANGING FROM L TO H
	DON'T CARE; ANY CHANGE PERMITTED	CHANGING; STATE UNKNOWN
	DOES NOT APPLY	CENTER LINE IS HIGH IMPEDANCE "OFF" STATE

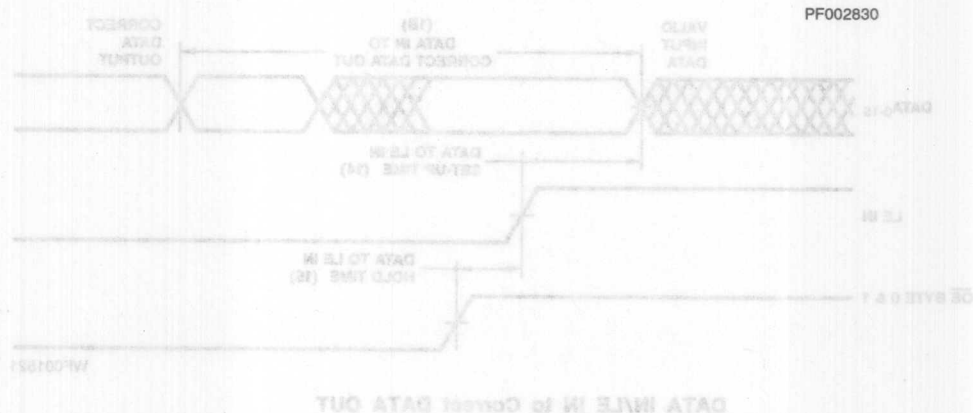
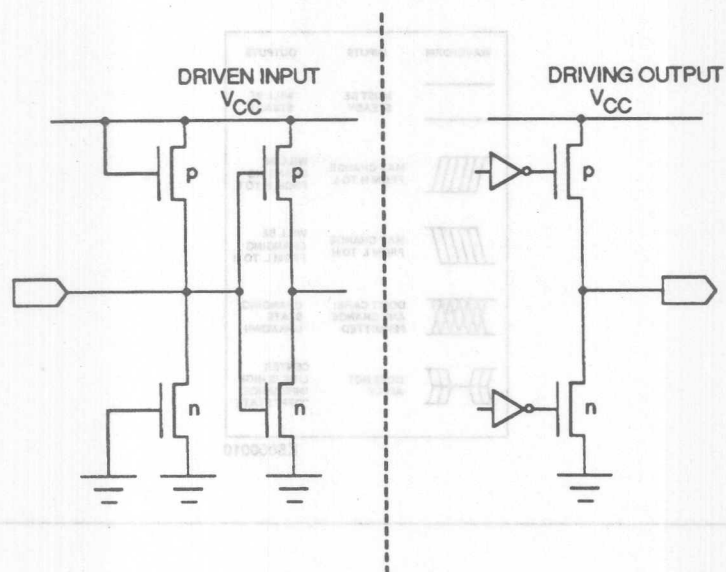
KS000010



WF001521

DATA IN/LE IN to Correct DATA OUT

EQUIVALENT INPUT/OUTPUT CIRCUIT DIAGRAMS



PF002830

Am29C660/A/B/C/D/E

CMOS Cascadable 32-Bit Error Detection and Correction Circuit

**Advanced
Micro
Devices**

DISTINCTIVE CHARACTERISTICS

- **Improves memory reliability**
 - Corrects all single-bit errors. Detects all double- and some triple-bit errors
- **Very high-speed error detection and correction**
 - Down to 9 ns data-in to error detection
- **Low-power CMOS process**
- **Cascadable for data words up to 64 bits**
- **Simplified byte operations**
 - Separate byte enables on the Data Output Latch for fast byte writes
- **Built-in diagnostics**
 - Proper EDC operation can be verified by the CPU via software control
- **Detects gross error conditions of all 1's or all 0's**

GENERAL DESCRIPTION

The Am29C660 CMOS Cascadable 32-Bit Error Detection and Correction Circuit (EDC) contains the logic necessary to generate check bits on a 32-bit data field according to a modified Hamming Code, and to correct the data word when check bits are supplied. Operating on data read from memory, the Am29C660 detects and corrects all single-bit errors and detects all double- and some triple-bit errors. For 32-bit words, 7 check bits are used.

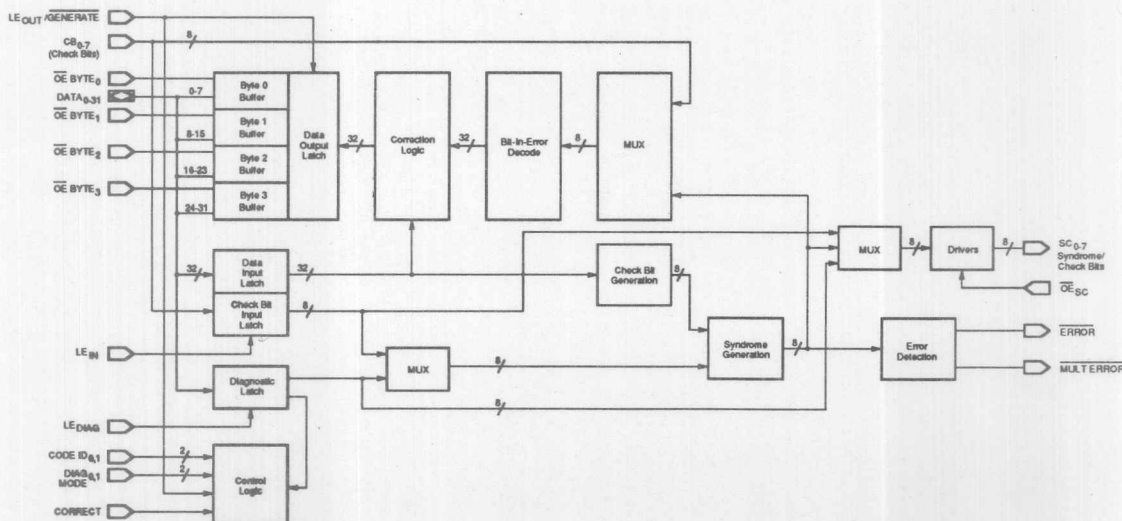
The Am29C660 is expandable to operate on 64-bit data words (8 check bits). In both configurations, the device

makes error syndromes available on separate outputs for error logging.

The Am29C660 also features two diagnostic modes in which diagnostic data can be forced into portions of the chip to simplify device testing and to execute system diagnostic functions.

When used with the Am29C668 Dynamic Memory Controller, the Am29C660 can perform AMD's invented memory "scrubbing" operation to provide highest data integrity.

BLOCK DIAGRAM



10565-001A

RELATED AMD PRODUCTS

Part No.	Description
Am29C668	4M Configurable Dynamic Memory Controller/Driver
Am29C983A	9-Bit x 4-Port Multiple Bus Exchange, High Speed
Am29C985	9-Bit x 4-Port Multiple Bus Exchange w/Parity
Am29C60A	16-Bit Cascadable EDC, High Speed
Am29C676	11-Bit Driver for 4M x 1 and 4M x 4 DRAMs
Am2965/6	8-Bit DRAM Driver (Inverting, Non-inverting)

- Very high-speed error detection and correction
- Down to 9 ns data-in to error detection
- Low-power CMOS process
- Cascadable for data words up to 64 bits
- Detects gross error conditions of all 1's or 0's
- Proper EDC operation can be verified by the CPU via software control
- Built-in diagnostics
- Latch for last byte writes

When used with the Am29C688 Dynamic Memory Controller, the Am29C680 can perform AMD's patented memory "scrubbing" operation to provide highest data integrity.

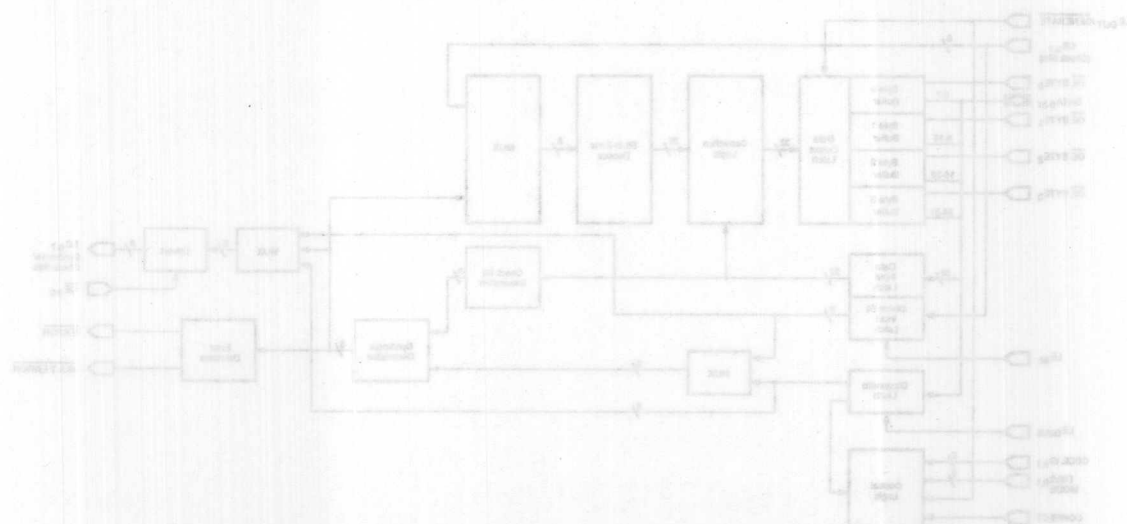
The Am29C680 also features two diagnostic modes in which diagnostic data can be forced into portions of the chip to simplify device testing and to execute system-diagnostic functions.

Am29C680 also features error syndromes available on separate outputs for error logging.

The Am29C680 CMOS Cascadable 32-Bit Error Detection and Correction Circuit (EDC) contains the logic necessary to generate check bits on a 32-bit data field according to a modified Hamming Code, and to correct the data word when check bits are supplied. Operating on data read from memory, the Am29C680 detects and corrects all single-bit errors and detects all double- and some triple-bit errors. For 32-bit words, 7 check bits are used.

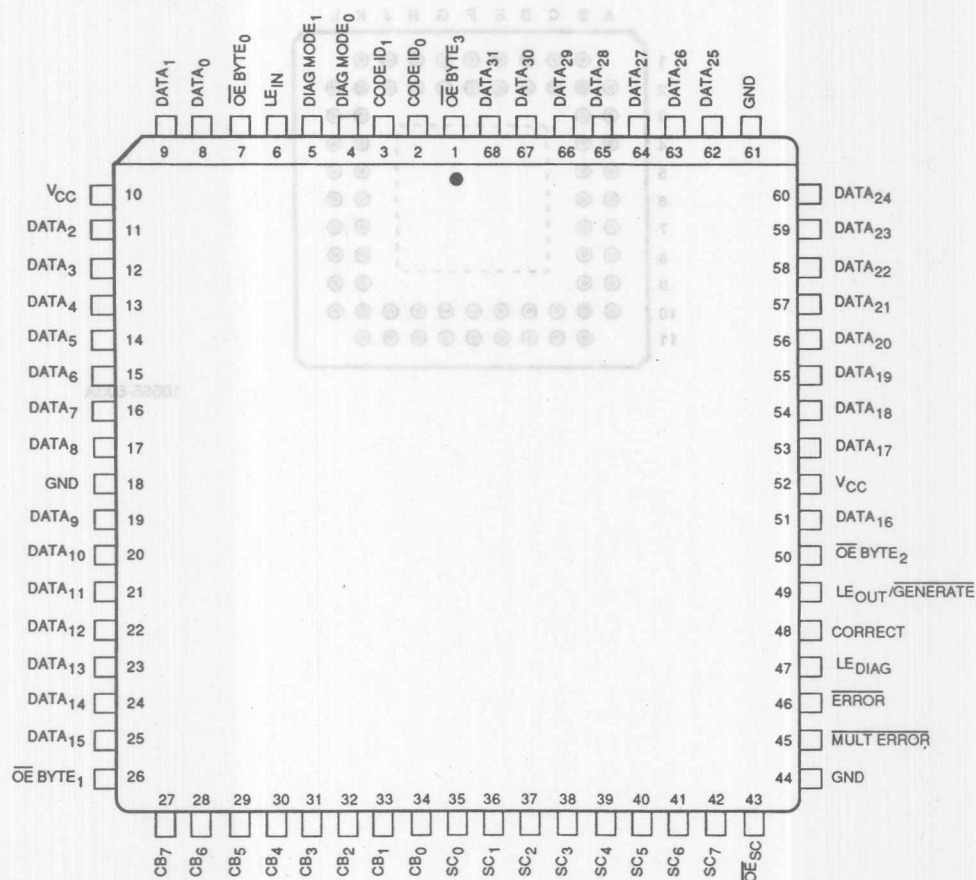
The Am29C680 is expandable to operate on 64-bit data words (8 check bits). In both configurations, the device

BLOCK DIAGRAM



CONNECTION DIAGRAMS

Top View



Note: Pin 1 is marked for orientation (PLCC only).

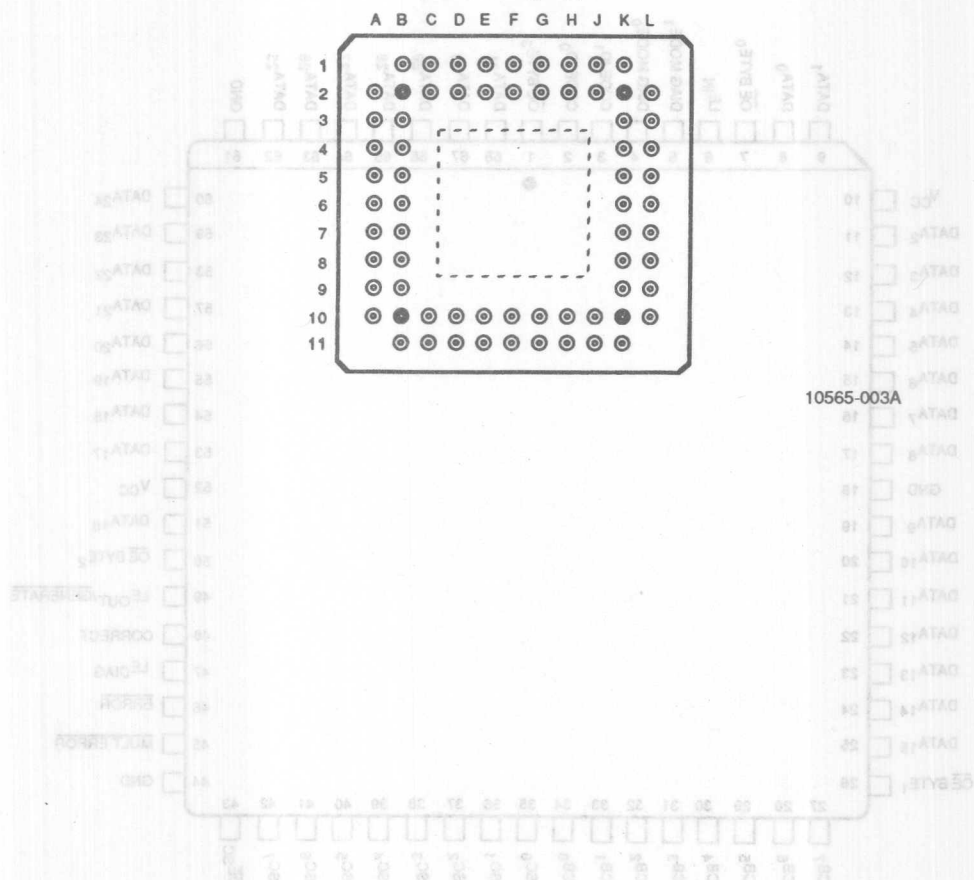
10565-002A

CONNECTION DIAGRAMS (Continued)

Bottom View

PGA

(Pins facing up)



10565-003A

Note: Pin 1 is marked for orientation (PUC only).

PGA PIN DESIGNATIONS

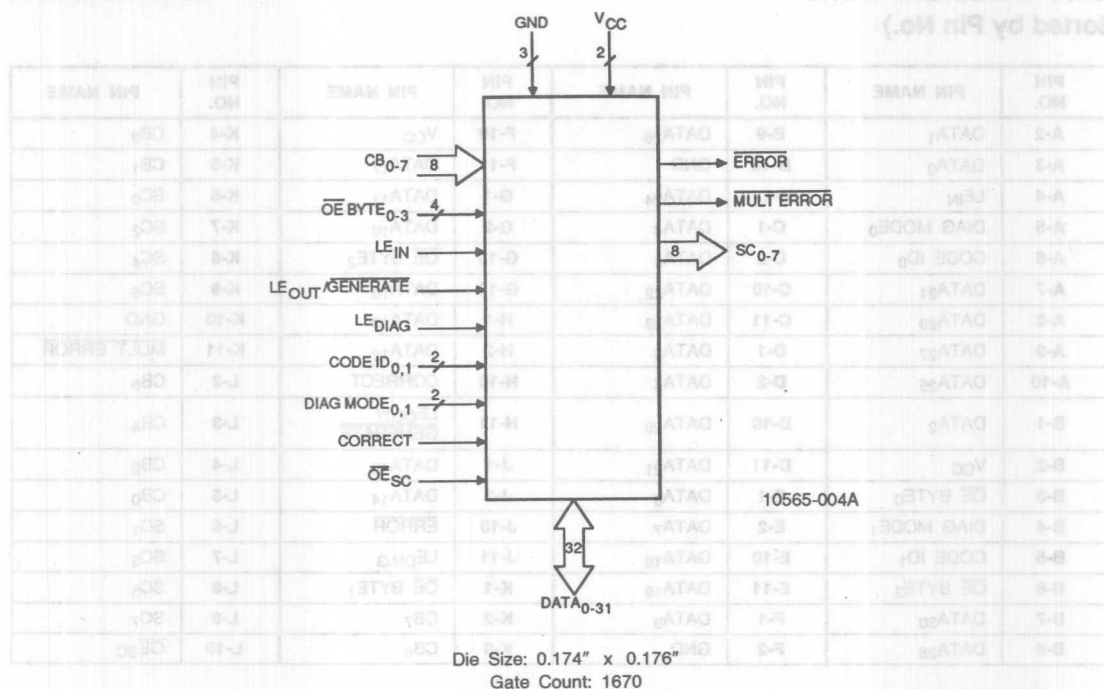
(Sorted by Pin No.)

PIN NO.	PIN NAME	PIN NO.	PIN NAME	PIN NO.	PIN NAME	PIN NO.	PIN NAME
A-2	DATA ₁	B-9	DATA ₂₆	F-10	V _{CC}	K-4	CB ₃
A-3	DATA ₀	B-10	GND	F-11	DATA ₁₇	K-5	CB ₁
A-4	LE _{IN}	B-11	DATA ₂₄	G-1	DATA ₁₁	K-6	SC ₀
A-5	DIAG MODE ₀	C-1	DATA ₄	G-2	DATA ₁₀	K-7	SC ₂
A-6	CODE ID ₀	C-2	DATA ₃	G-10	OE BYTE ₂	K-8	SC ₄
A-7	DATA ₃₁	C-10	DATA ₂₂	G-11	DATA ₁₆	K-9	SC ₆
A-8	DATA ₂₉	C-11	DATA ₂₃	H-1	DATA ₁₃	K-10	GND
A-9	DATA ₂₇	D-1	DATA ₆	H-2	DATA ₁₂	K-11	MULT ERROR
A-10	DATA ₂₅	D-2	DATA ₅	H-10	CORRECT	L-2	CB ₆
B-1	DATA ₂	D-10	DATA ₂₀	H-11	LE _{OUT} / GENERATE	L-3	CB ₄
B-2	V _{CC}	D-11	DATA ₂₁	J-1	DATA ₁₅	L-4	CB ₂
B-3	OE BYTE ₀	E-1	DATA ₈	J-2	DATA ₁₄	L-5	CB ₀
B-4	DIAG MODE ₁	E-2	DATA ₇	J-10	ERROR	L-6	SC ₁
B-5	CODE ID ₁	E-10	DATA ₁₈	J-11	LE _{DIAG}	L-7	SC ₃
B-6	OE BYTE ₃	E-11	DATA ₁₉	K-1	OE BYTE ₁	L-8	SC ₅
B-7	DATA ₃₀	F-1	DATA ₉	K-2	CB ₇	L-9	SC ₇
B-8	DATA ₂₈	F-2	GND	K-3	CB ₅	L-10	OE _{SC}

(Sorted by Pin Name)

PIN NO.	PIN NAME	PIN NO.	PIN NAME	PIN NO.	PIN NAME	PIN NO.	PIN NAME
L-5	CB ₀	D-1	DATA ₆	C-11	DATA ₂₃	H-11	LE _{OUT} / GENERATE
K-5	CB ₁	E-2	DATA ₇	B-11	DATA ₂₄	K-11	MULT ERROR
L-4	CB ₂	E-1	DATA ₈	A-10	DATA ₂₅	B-3	OE BYTE ₀
K-4	CB ₃	F-1	DATA ₉	B-9	DATA ₂₆	K-1	OE BYTE ₁
L-3	CB ₄	G-2	DATA ₁₀	A-9	DATA ₂₇	G-10	OE BYTE ₂
K-3	CB ₅	G-1	DATA ₁₁	B-8	DATA ₂₈	B-6	OE BYTE ₃
L-2	CB ₆	H-2	DATA ₁₂	A-8	DATA ₂₉	L-10	OE _{SC}
K-2	CB ₇	H-1	DATA ₁₃	B-7	DATA ₃₀	K-6	SC ₀
A-6	CODE ID ₀	J-2	DATA ₁₄	A-7	DATA ₃₁	L-6	SC ₁
B-5	CODE ID ₁	J-1	DATA ₁₅	A-5	DIAG MODE ₀	K-7	SC ₂
H-10	CORRECT	G-11	DATA ₁₆	B-4	DIAG MODE ₁	L-7	SC ₃
A-3	DATA ₀	F-11	DATA ₁₇	J-10	ERROR	K-8	SC ₄
A-2	DATA ₁	E-10	DATA ₁₈	B-10	GND	L-8	SC ₅
B-1	DATA ₂	E-11	DATA ₁₉	F-2	GND	K-9	SC ₆
C-2	DATA ₃	D-10	DATA ₂₀	K-10	GND	L-9	SC ₇
C-1	DATA ₄	D-11	DATA ₂₁	J-11	LE _{DIAG}	B-2	V _{CC}
D-2	DATA ₅	C-10	DATA ₂₂	A-4	LE _{IN}	F-10	V _{CC}

LOGIC SYMBOL



Package Information

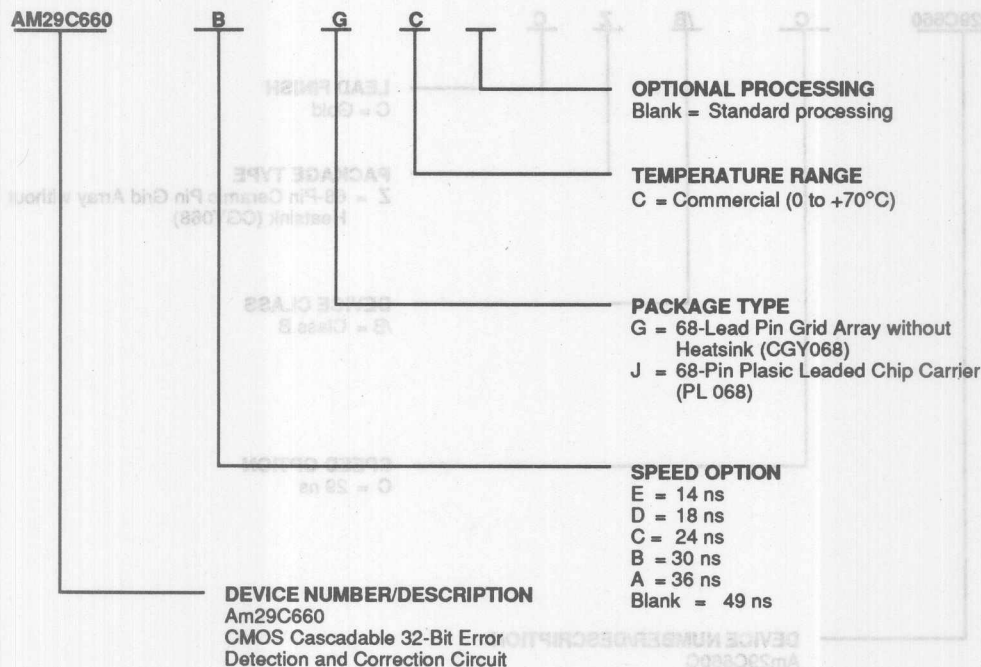
Parameter	PGA	PLCC	Unit
θ_{JA}	34	35	°C/W
θ_{JC}	-	N/A	°C/W

ORDERING INFORMATION

Standard Products

AMD standard products are available in several packages and operating ranges. The order number (Valid Combination) is formed by a combination of the following elements:

Device Number
Speed Option (If applicable)
Package Type
Temperature Range
Optional Processing



Valid Combinations	
AM29C660	GC, JC
AM29C660A	
AM29C660B	
AM29C660C	
AM29C660D	
AM29C660E	

Valid Combinations

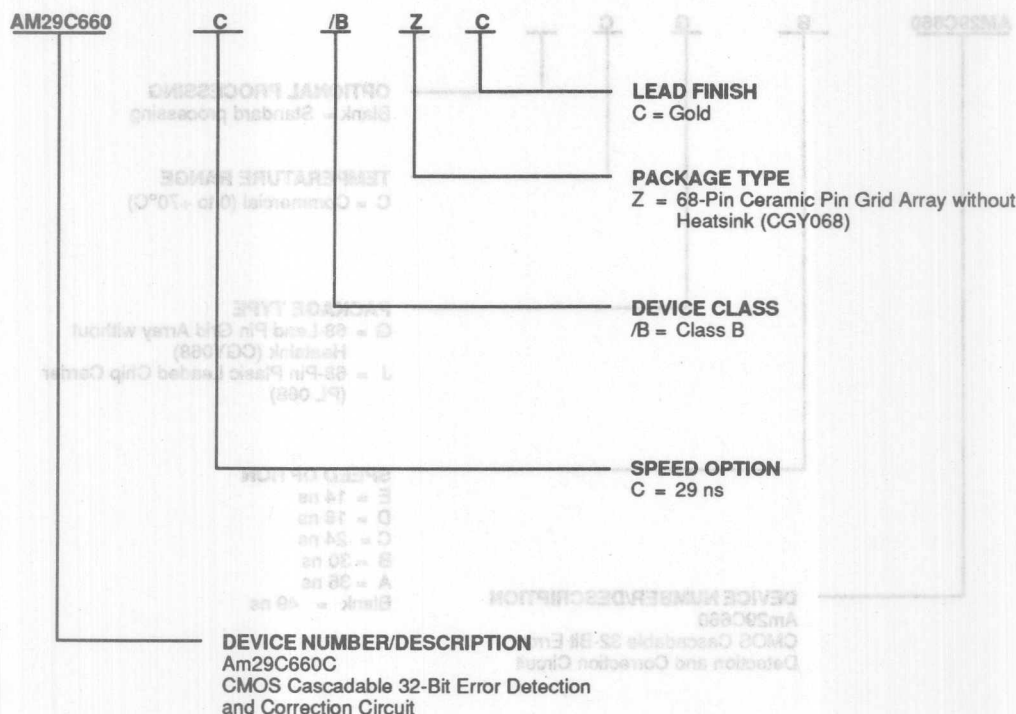
Valid Combinations list configurations planned to be supported in volume for this device. Consult the local AMD sales office to confirm availability of specific valid combinations or to check on newly released combinations, and to obtain additional data on AMD's standard military grade products.

MILITARY ORDERING INFORMATION

APL Products

AMD products for Aerospace and Defense applications are available in several packages and operating ranges. APL (Approved Products List) products are fully compliant with MIL-STD-883C requirements. The order number (Valid Combination) is formed by a combination of the following elements:

Device Number
Speed Option (If applicable)
Device Class
Package Type
Lead Finish



Valid Combinations	
AM29C660C	/BZC

Valid Combinations

Valid Combinations list configurations planned to be supported in volume for this device. Consult the local AMD sales office to confirm availability of specific valid combinations or to check on newly released combinations, and to obtain additional data on AMD's standard military grade products.

Group A Tests

Group A tests consist of Subgroups
1, 2, 3, 7, 8, 9, 10, 11.

PIN DESCRIPTION

CB₀₋₇

Check Bits (Inputs)

These eight check bit lines are used to input bits for error detection. They are also used to input syndrome bits for error correction in the 64-bit configuration.

CODE ID_{0,1}

Code Identification (Inputs)

These two code identification inputs identify the size of the total data word to be processed. The two allowable data word sizes are 32 and 64 bits and their respective modified Hamming Codes are designated 32/39 and 64/72. The CODE ID inputs are also used to instruct the EDC that the CODE ID_{0,1}, DIAG MODE_{0,1}, and CORRECT signals are to be taken from the Diagnostic Latch, rather than from the input control lines (Reference Table 1).

CORRECT

Correct (Input; Active HIGH)

This signal allows the correction network to correct any single-bit error in the Data Input Latch before putting it into the Data Output Latch by complementing the bit-in-error. When the signal is LOW, the EDC routes data directly from the Data Input Latch to the Data Output Latch without correction.

DATA₀₋₃₁

Data (Inputs/Outputs; Three-State)

These bidirectional data lines provide input to the Data Input Latch and Diagnostic Latch, and receive output from the Data Output Latch. DATA₀ is the least significant bit and DATA₃₁ is the most significant bit.

DIAG MODE_{0,1}

Diagnostic Mode Select (Inputs; Active HIGH)

These two lines control the initialization and diagnostic operation of the EDC circuit (Reference Table 2).

ERROR

Error Detection Flag (Output; Active LOW)

When the EDC is in the Detect or Detect/Correct Mode, this output goes LOW if one or more syndrome bits are nonzero, indicating one or more data and/or check bits are in error. If all syndrome bits are zero, there are no errors detected and the output will be HIGH. In the Generate Mode, $\overline{\text{ERROR}}$ is forced HIGH.

GND (3)

0-V Power Supply

These pins are the 0-V power supply to the EDC circuit. All grounds must be connected during device operation.

LE_{DIAG}

Diagnostic Latch Enable (Input)

This is the latch enable for the Diagnostic Latch. When HIGH, the Diagnostic Latch follows the 32-bit data on

the input lines. When LOW, the outputs of the Diagnostic Latch are latched to their previous states. The Diagnostic Latch holds diagnostic check bits and internal control signals for CODE ID_{0,1}, DIAG MODE_{0,1}, and CORRECT.

LE_{IN}

Latch Enable Data Input Latch (Input)

This input controls latching of the input data. When HIGH, the Data Input Latch and Check Bit Input Latch follow the input data and input check bits, respectively. When LOW, the Data Input Latch and Check Bit Input Latch are latched to their previous states.

LE_{OUT}/GENERATE

Latch Enable – Data Output Latch (Input)/Generate Check Bits (Input; Active LOW)

This is a multifunction pin. When it is LOW, it operates in the Check Bit Generate Mode. In this mode, the device generates the check bits or partial check bits specific to the data in the Data Input Latch. The generated check bits are placed on the syndrome/check bit outputs. The Data Output Latch is latched to its previous state when this pin is LOW.

When HIGH, the device is in the Detect or Detect/Correct Mode. In the Detect Mode, the device detects single and multiple errors, and generates syndrome bits specific to the data in the Data Input Latch and Check Bit Input Latch. In the Detect/Correct Mode, single-bit errors are automatically corrected, with the corrected data placed at the inputs of the Data Output Latch. The syndrome result is placed on the syndrome/check bit outputs and indicates, in a coded form, the number of errors and the specific bit in error. When HIGH, the Data Output Latch follows the output of the Data Input Latch as modified by the correction logic network.

In the Detect/Correct Mode, single-bit errors are corrected by the network before being loaded into the Data Output Latch. In the Detect Mode, the contents of the Data Input Latch are passed through the correction network unchanged into the Data Output Latch. The Data Output Latch is disabled with its contents unchanged if the EDC is in the Generate Mode.

MULT ERROR

Multiple Error Detection Flag (Output; Active LOW)

When LOW in the Detect or Detect/Correct Mode, this output indicates that two or more bit errors have been detected. If HIGH, either one or no errors have been detected. In the Generate Mode, $\overline{\text{MULT ERROR}}$ is forced HIGH.

OE BYTE₀₋₃

Output Enable Bytes 0-3 (Inputs; Active LOW)

These lines control the three-state output buffers for each of the four bytes of the Data Output Latch. When LOW, they enable the output buffer of the Data Output

Latch. When HIGH, they force the Data Output Latch buffer into the high impedance state. Any number of bytes of the Data Output Latch is easily activated by separately selecting any of the four enable lines.

OE_{sc} Output Enable, Syndrome/Check Bits (Input; Active LOW)

When in the LOW state, the three-state output lines SC₀₋₇ are enabled. When this input is HIGH, the syndrome/check bit outputs are in the high-impedance state.

SC_{0,7} Syndrome/Check Bits (Outputs; Three-State)

These eight three-state outputs contain the check/partial check bits when the EDC is in the Generate Mode.

They also contain the syndrome/partial-syndrome bits when the device is in the Detect or Detect/Correct Modes.

V_{cc} (2) + 5-V Positive Power Supply Voltage

These pins are the positive + 5-V power supply to the EDC Circuit. All V_{cc} pins must be connected during device operation.

This is a multifunction pin. When it is LOW, it operates in the Check Bit Generate Mode. In this mode, the device generates the check bits or partial check bits specific to the data in the Data Input Latch. The generated check bits are placed on the syndrome/check bit outputs. The Data Output Latch is latched to its previous state when this pin is LOW.

When HIGH, the device is in the Detect or Detect/Correct Mode. In the Detect Mode, the device detects single and multiple errors, and generates syndrome bits specific to the data in the Data Input Latch and Check Bit Input Latch. In the Detect/Correct Mode, single-bit errors are automatically corrected, with the corrected data placed at the inputs of the Data Output Latch. The syndrome result is placed on the syndrome/check bit outputs and indicates, in a coded form, the number of errors and the specific bit in error. When HIGH, the Data Output Latch follows the output of the Data Input Latch as modified by the correction logic network.

In the Detect/Correct Mode, single-bit errors are corrected by the network before being loaded into the Data Output Latch. In the Detect Mode, the contents of the Data Input Latch are passed through the correction network unchanged into the Data Output Latch. The Data Output Latch is disabled with its contents unchanged if the EDC is in the Generate Mode.

MULTI ERROR Multiple Error Detection Flag (Output; Active LOW)

When LOW in the Detect or Detect/Correct Mode, this output indicates that two or more bit errors have been detected. If HIGH, either one or no errors have been detected. In the Generate Mode, MULTIPLE ERROR is forced HIGH.

OE BYTES Output Enable Bytes 0-3 (Input; Active LOW)

These lines control the three-state output buffers for each of the four bytes of the Data Output Latch. When LOW, they enable the output buffer of the Data Output

CORRECT Correct (Input; Active HIGH)

This signal allows the correction network to correct any single-bit error in the Data Input Latch before putting it into the Data Output Latch by complementing the bit-in-error. When the signal is LOW, the EDC routes data directly from the Data Input Latch to the Data Output Latch without correction.

DATA₀₋₃₁
Data (Input/Output; Three-State)
These bidirectional data lines provide input to the Data Input Latch and Diagnostic Latch, and receive output from the Data Output Latch. DATA is the least significant bit and DATA₃₁ is the most significant bit.

DIAG MODE₀₋₁
Diagnostic Mode Select (Input; Active HIGH)
These two lines control the initialization and diagnostic operation of the EDC circuit (Reference Table 2).

ERROR Error Detection Flag (Output; Active LOW)

When the EDC is in the Detect or Detect/Correct Mode, this output goes LOW if one or more syndrome bits are nonzero, indicating one or more data and/or check bits are in error. If all syndrome bits are zero, there are no errors detected and the output will be HIGH. In the Generate Mode, ERROR is forced HIGH.

GND (3) 0-V Power Supply

These pins are the 0-V power supply to the EDC circuit. All grounds must be connected during device operation.

LEADS Diagnostic Latch Enable (Input)

This is the latch enable for the Diagnostic Latch. When HIGH, the Diagnostic Latch follows the 32-bit data on

FUNCTIONAL DESCRIPTION

EDC Architecture

The Am29C660 EDC Circuit is a powerful 32-bit cascable slice used for check bit generation, error detection, error correction, and diagnostics.

As shown in the block diagram, the device consists of the following:

- Data Input Latch
- Check Bit Input Latch
- Check Bit Generation Logic
- Syndrome Generation Logic
- Error Detection Logic
- Error Correction Logic
- Data Output Latch and Output Buffers
- Diagnostics Latch
- Control Logic

Data Input Latch

The Latch Enable Input, LE_{IN} , controls the loading of 32 bits of data into the Data Input Latch. Depending upon the control mode, the input data is either used for check bit generation or error detection/correction.

Check Bit Input Latch

Eight check bits are loaded under the control of LE_{IN} . Check bits are used in the Error Detection and Error Detection/Correction Modes.

Check Bit Generation Logic

This block generates the appropriate check bits for the 32 bits of data in the Data Input Latch. The check bits are generated according to a modified Hamming Code.

Syndrome Generation Logic

In both the Error Detection and Error Detection/Correction Modes, this logic block compares the check bits read from the memory against a newly generated set of check bits produced for the data read in from the memory. If both sets of check bits match, then there are no errors. If there is a mismatch, then one or more of the data and/or check bits is in error.

The syndrome bits are produced by an Exclusive-OR of the two sets of check bits. If the two sets of check bits are identical (meaning there are no errors), the syndrome bits will be all zeros. If there is a single-bit error, the syndrome bits can be decoded to determine the bit-in-error.

This logic block decodes the syndrome bits generated by the Syndrome Generation Logic. If there are no errors in either the input data or check bits, the **ERROR** and **MULT ERROR** outputs remain HIGH. If one or more errors are detected, **ERROR** goes LOW. If two or more errors are detected, both **ERROR** and **MULT ERROR** go LOW.

Error Correction Logic

For single-bit errors, the Error Correction Logic corrects (by complementing) the single data bit in error. This corrected data is loaded into the Data Output Latch, which can then be read onto the bidirectional data lines. If the single-bit error is one of the check bits, the correction logic does not place corrected check bits on the syndrome/check bit outputs. If the corrected check bits are needed, the EDC must be switched to the Generate Mode.

Data Output Latch and Output Buffers

The Data Output Latch is used for storing the result of an error correction operation. The latch is loaded from the correction logic under control of the Data Output Latch Enable, $LE_{OUT}/GENERATE$. The Data Output Latch may also be directly loaded from the Data Input Latch while in the Pass-Thru Mode.

Four data bytes in the Data Output Latch may be end-independently (by $OE_{BYTE0-3}$) for reading onto the bidirectional data bus. This feature facilitates byte operations.

Diagnostics Latch

This is a 32-bit latch loadable from the bidirectional data lines under the control of the Diagnostic Latch Enable, LE_{DIAG} . The Diagnostic Latch contains check bit information in one byte and control information in the other bytes. The Diagnostic Latch is used for driving the device when in the Internal Control Mode, or for supplying check bits when in one of the Diagnostic Modes.

Control Logic

The Control Logic specifies the mode in which the EDC is operating. Normally, the control logic is driven by the external control inputs. However, in the Internal Control Mode, the control signals are taken from the Diagnostic Latch. Since LE_{OUT} and $GENERATE$ are controlled by the same pin, the latching action (LE_{OUT} from HIGH to LOW) of the Data Output Latch causes the EDC to go into the Generate Mode.

Detailed Operational Description

The Am29C660 contains the logic necessary to generate check bits on a 32-bit data field according to a modified Hamming Code. Operating on data read from memory, the EDC will correct any single-bit error, and will detect all double and some triple-bit errors. The Am29C660 may be configured to operate on 32-bit data words (with 7 check bits) and 64-bit data words (with 8 check bits). In either configuration, the device makes the error syndrome bits available on separate outputs for error logging.

Code and Byte Specification

The EDC Circuit may be configured in several different ways, and operates differently in each configuration. It is necessary to indicate to the device what size data word is involved and which bytes of the data word it is processing.

This is done with the input lines CODE ID_{0,1} as shown in Table 1. The two modified Hamming codes referred to in Table 1 are:

- 32/39 Code: 32 data bits, 7 check bits (39 bits in total)
- 64/72 Code: 64 data bits, 8 check bits (72 bits in total)

Control Mode Selection

There are nine operating modes of the Am29C660. Eight of these modes are selected as shown in Tables 2 and 3. Table 2 is the Diagnostic Mode Control Decode Table, and Table 3 is the Mode Control Decode Table. The Diagnostic Mode pins, DIAG MODE_{0,1}, define the five basic areas of operation. GENERATE and CORRECT further divide the operations into eight functions. The ninth mode is the Internal Control Mode which is selected by the CODE ID inputs as shown in Table 1.

Table 1. CODE ID Decode

CODE ID ₁	CODE ID ₀	Hamming Code and Slice Selected
0	0	Code 32/39, 32-Bit Data Word
0	1	Internal Control Mode
1	0	Code 64/72, Lower 32-Bit Slice (0-31)
1	1	Code 64/72, Upper 32-Bit Slice (32-63)

Table 2. Diagnostic Mode Control Decode

DIAG MODE ₁	DIAG MODE ₀	CORRECT	Diagnostic Mode Selected
0	0	X	Normal EDC Function Mode or Non-Diagnostic Mode
0	1	X	Diagnostic Generate Mode
1	0	X	Diagnostic Detect Mode and Diagnostic Detect/Correct Mode
1	1	1	Initialize Mode
1	1	0	Pass-Thru Mode

Table 3. Mode Control Decode

Operating Mode	DIAG MODE 1 0	LEout/GENERATE	CORRECT	Contents of Data Output Latch	SC ₀₋₇ and OEsc = LOW	ERROR and MULT ERROR
Generate	0 0 1 0	0	X		Check bits generated from Data Input Latch	High
Detect	0 0 0 1	1	0	Data Input Latch	Generated from Data Input/Check Bit Latches	Valid
Detect/Correct	0 0 0 1	1	1	Data Input Latch with single bit error detected	Generated from Data Input/Check Bit Latches	Valid
Pass-Thru	1 1	1 or ↓	0	Data Input Latch	Check Bit Latch	HIGH
Diagnostic Generate	0 1	0	X	—	Check bits from Diagnostic Latch	—
Diagnostic Detect	1 0	1	0	Data Input Latch	Data Input check bits from Diagnostic Latch	Valid
Diagnostic Detect/Correct	1 0	1	1	Data Input Latch with single-bit error detected	Data Input check bits from Diagnostic Latch	Valid
Initialize	1 1	1 or ↓	1	Data Input Latch set to all "0"s	Check bits generated from Data Input Latch	—
Internal Control	CODE ID _{0,1} = 10 Control Signals CODE ID _{0,1} DIAG MODE _{0,1} and CORRECT are taken from the Diagnostic Latch					

* In the Generate Mode, data is read into the EDC circuit and the check bits are generated. The same data is written to the memory along with the check bits. Since the Data Output Latch is not used in the Generate Mode, LEout, being LOW (since it is tied to Generate) does not affect the writing of check bits.

The Generate Mode is used to display the check bits on the SC₀₋₇ outputs. The Error Detect Mode provides an indication of an error or multiple errors on the ERROR and MULTERROR outputs. Single-bit errors are not corrected in this mode. The syndrome bits are provided on the SC₀₋₇ outputs. In the Diagnostic Detect Mode, the syndrome bits are generated by comparing the internally generated check bits from the Data Input Latch with check bits stored in the Diagnostic Latch (as opposed to those in the Check Bit Latch).

The Detect/Correct Mode is similar to the Detect Mode except that single-bit errors will be corrected (by complementing) and made available as input to the Data Output Latch. In the Diagnostic Detect/Correct Mode, single-bit errors will be corrected as determined by the syndrome bits, which are in turn generated from the check bits corresponding to the data in the Data Input Latch and the check bits in the Diagnostic Latch.

In the Initialize Mode check bits are generated for all zero data bits. The Data Input Latch is held to a logic zero and is made available as input to the Data Output Latch.

In the Internal Control Mode, the control signals CODE ID_{0,1}, DIAG MODE_{0,1}, and CORRECT are taken from the Diagnostic Latch and their respective control inputs are disregarded.

Check and Syndrome Bits

The Am29C660 provides either check or syndrome bits on the three-state outputs SC₀₋₇. Check bits are generated from the Data Input Bits. Syndrome bits are an Exclusive-OR of the check bits generated from the data read from the memory and the check bits read from the memory with the stored data.

Syndrome bits can be decoded to determine the single bit-in-error or to indicate a double- or triple-bit error.

The check bits generated by the Am29C660 are designated as follows:

- 32-bit configuration: CX, C0, C1, C2, C4, C8, C16
- 64-bit configuration: CX, C0, C1, C2, C4, C8, C16, C32
- Pin Name: SC₀, SC₁, SC₂, SC₃, SC₄, SC₅, SC₆, SC₇.

The syndrome bits generated by the Am29C660 are designated as follows:

- 32-bit configuration: SX, S0, S1, S2, S4, S8, S16
- 64-bit configuration: SX, S0, S1, S2, S4, S8, S16, S32
- Pin Name: SC₀, SC₁, SC₂, SC₃, SC₄, SC₅, SC₆, SC₇.

32-Bit Word Configuration

Data Field Format

The 32-bit format consists of 32 data bits and 7 check bits and is referred to as the 32/39 code. The format is shown in Figure 1.

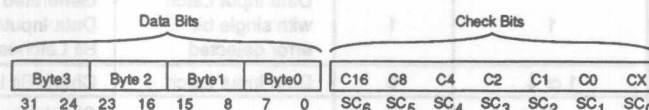


Figure 1. 32-Bit Data Word Format

Chip Configuration

A single Am29C660 EDC Circuit connected as shown in Figure 2 is all that is necessary to perform single-bit error correction and double-bit error detection on a 32-bit

data field. In this configuration, only seven check bits are required. Therefore, CB₇ is a "don't care" and CODE ID_{0,1} = 00.

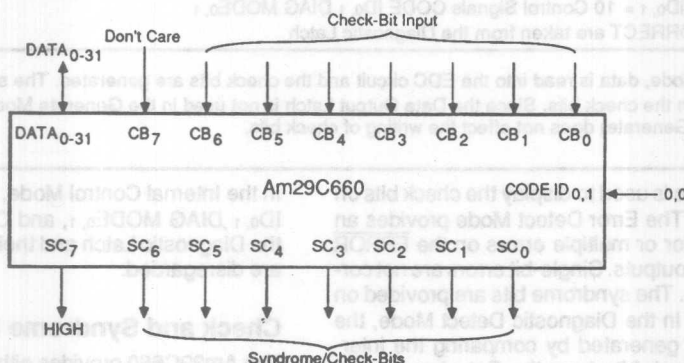


Figure 2. 32-Bit Configuration

Generate Mode

In this mode, check bits will be generated that correspond to the contents of the Data Input Latch. The check bits generated are placed on the outputs SC₀₋₆. SC₇ is a logic "1" or HIGH.

Check bits are generated according to a modified Hamming Code. Details of the code for check bit generation

are shown in Tables 4-1 and 4-2. Check bits are generated as either an XOR or XNOR of 16 of the 32 bits as indicated in the table. The XOR function results in an even parity check bit, the XNOR in an odd parity check bit.

Figure 3 shows the data flow in the Generate Mode.

**Table 4-1. 32-Bit Configuration Check Bit Encoding
(Data Bits 0–15)**

Generated Check Bits	Parity	Participating Data Bits															
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CX	Even (XOR)	X				X		X	X	X	X	X			X		
C ₀	Even (XOR)	X	X	X		X		X		X		X		X			
C ₁	Odd (XNOR)	X		X	X			X		X	X			X		X	
C ₂	Odd (XNOR)	X	X			X	X	X				X	X	X			
C ₄	Even (XOR)		X	X		X	X	X	X						X	X	
C ₈	Even (XOR)									X	X	X	X	X	X	X	X
C ₁₆	Even (XOR)	X	X	X	X	X	X	X	X								

**Table 4-2. 32-Bit Configuration Check Bit Encoding
(Data Bits 16–31)**

Generated Check Bits	Parity	Participating Data Bits															
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
CX	Even (XOR)		X	X	X		X				X			X	X	X	X
C ₀	Even (XOR)	X	X	X		X		X		X		X		X			
C ₁	Odd (XNOR)	X		X	X		X		X	X	X			X		X	
C ₂	Odd (XNOR)	X	X			X	X	X				X	X	X			
C ₄	Even (XOR)		X	X		X	X	X	X						X	X	
C ₈	Even (XOR)									X	X	X	X	X	X	X	X
C ₁₆	Even (XOR)									X	X	X	X	X	X	X	X

The check bit is generated as either an XOR or XNOR of the sixteen data bits noted by an "X" in the table.

Detect Mode

In this mode the device will compare the check bits generated from the contents of the Data Input Latch with the contents of the Check Bit Latch, and will detect all single- and double-bit errors and some triple-bit errors. If one or more errors are detected, **ERROR** goes LOW. If two or more errors are detected, **MULT ERROR** and **ERROR** go LOW. Both error indicators are HIGH if there is no error.

The syndrome bits which are generated during error detection are available on the outputs SC_{0–6}. SC₇ remains HIGH. The syndrome bits may be decoded to determine if a bit error was detected and for a single-bit error, which of the data or check bits is in error. Table 5 shows the syndrome bit decoding for the 32-bit data word configuration. If no error is detected, the syndrome bits will all be zeros (except SC₇ which is tied to a logical "1").

Table 5. Syndrome Bit Decoding Matrix

Syndrome Bits	S16	0	1	0	1	0	1	0	1
S8	0	0	1	1	0	0	1	1	1
S4	0	0	0	0	1	1	1	1	1
SX S0 S1 S2									
0 0 0 0	*	C16	C8	T	C4	T	T	30	
0 0 0 1	C2	T	T	27	T	5	M	T	
0 0 1 0	C1	T	T	25	T	3	15	T	
0 0 1 1	T	M	13	T	23	T	T	M	
0 1 0 0	C0	T	T	24	T	2	M	T	
0 1 0 1	T	1	12	T	22	T	T	M	
0 1 1 0	T	M	10	T	20	T	T	M	
0 1 1 1	16	T	T	M	T	M	M	T	
1 0 0 0	CX	T	T	M	T	M	14	T	
1 0 0 1	T	M	11	T	21	T	T	M	
1 0 1 0	T	M	9	T	19	T	T	31	
1 0 1 1	M	T	T	29	T	7	M	T	
1 1 0 0	T	M	8	T	18	T	T	M	
1 1 0 1	17	T	T	28	T	6	M	T	
1 1 1 0	M	T	T	26	T	4	M	T	
1 1 1 1	T	0	M	T	M	T	T	M	

* = No errors detected

Number = The bit number of the single bit-in-error

T = Two errors detected

M = More than two errors detected

Detect/Correct Mode

In this mode, the EDC functions the same way as in the Detect Mode except that the correction network is enabled to correct, by complementing, any single-bit error in the Data Input Latch before placing the data on the inputs of the Data Output Latch. If a multiple error is detected, the output of the correction network is undefined. If a single-bit error occurs to a check bit, there is no automatic correction. If a check bit correction is desired, it can be done by placing the device in the Generate Mode to produce the correct check bit sequence for the data in the Data Input Latch.

Pass-Thru Mode

In this mode, the unmodified contents of the Data Input Latch are placed on the inputs of the Data Output Latch, and the contents of the Check bit Latch are placed on the output SC₀₋₆ (SC₇ is a logical "1"). ERROR and MULT ERROR are forced HIGH in this mode (inactive).

Diagnostic Generate Mode

This is one of the three special Diagnostic Modes selected by the control inputs DIAG MODE_{0,1}. This mode is similar to the normal Generate Mode except that the check bits are not generated from the contents of the Data Input Latch. They are instead taken directly from the contents of the Diagnostic Latch. Table 6 shows the Diagnostic Latch coding format. Figures 5 and 6 illustrate the flow of data during the two diagnostic modes.

Diagnostic Detect, Diagnostic Detect/Correct Modes

These two special Diagnostic Modes are also selected by the control inputs DIAG MODE_{0,1}. These modes are similar to the normal Detect and Detect/Correct Modes except that the check bits are taken from the Diagnostic Latch, rather than from the Check Bit Input Latch.

Initialize Mode

In this mode the inputs of the Data Output Latch are forced to all zeros. The syndrome bit outputs SC₀₋₆ are generated to correspond to the all-zero data on the Data Output Latch. SC₇ is tied high. The ERROR and MULT ERROR outputs are forced HIGH in this mode. The Initialize Mode is useful after power-up when RAM contents are random. The EDC may be placed in this mode and its outputs written into all memory addresses under processor control.

Internal Control Mode

This mode is selected by the external control inputs CODE ID₀₋₁. When in the Internal Control Mode, the Am29C660 takes the CODE ID₀₋₁, DIAG MODE₀₋₁, and CORRECT control signals from the Diagnostic Latch rather than from the external input lines.

Table 6. 32-Bit Diagnostic Latch Coding Format

Data Bit	Internal Function
0	Diagnostic Check Bit X
1	Diagnostic Check Bit 0
2	Diagnostic Check Bit 1
3	Diagnostic Check Bit 2
4	Diagnostic Check Bit 4
5	Diagnostic Check Bit 8
6	Diagnostic Check Bit 16
7	Don't Care
8	CODE ID ₀
9	CODE ID ₁
10	DIAG MODE ₀
11	DIAG MODE ₁
12	CORRECT
13-31	Don't Care

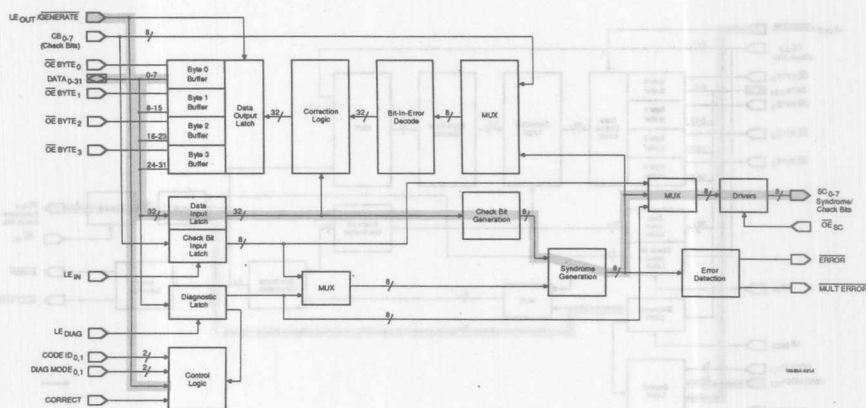


Figure 3. Check Bit Generation Data Path

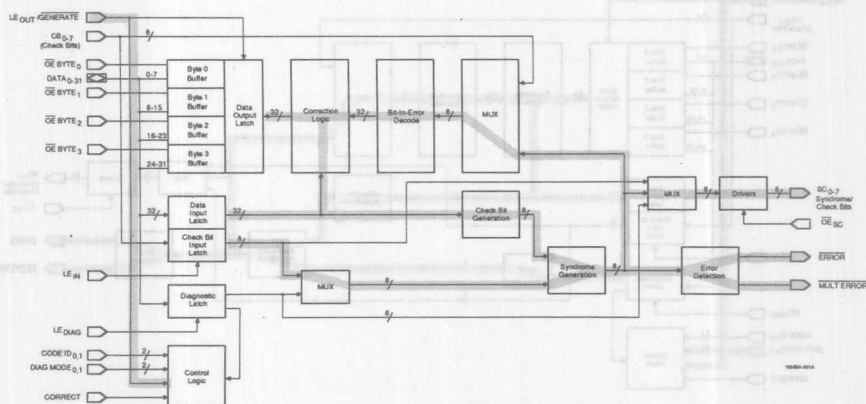


Figure 4. Error Detection and Correction Data Path

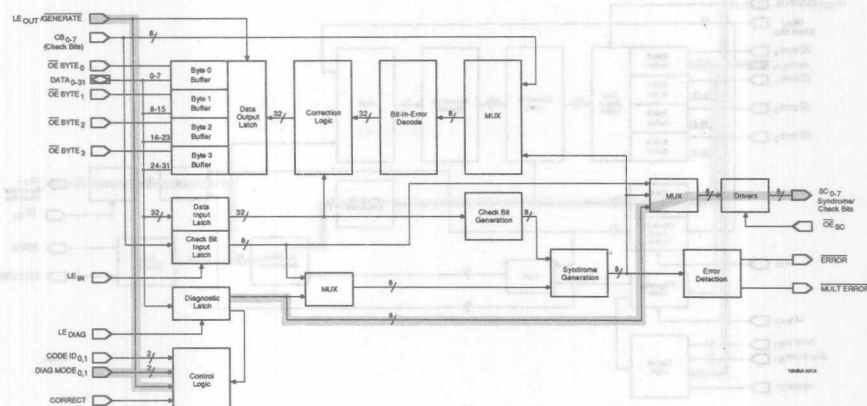


Figure 5. Diagnostic Check Bit Generation Data Path

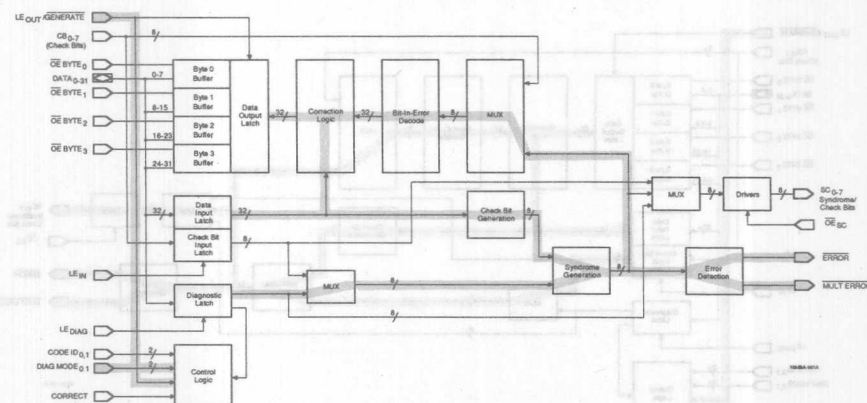


Figure 6. Diagnostic Detect and Correct Data Path

64-Bit Data Word Configuration

Data Field Format

The 64-bit format consists of 64 data bits and 8 check bits and is referred to as the 64/72 code. The format is shown in Figure 7.

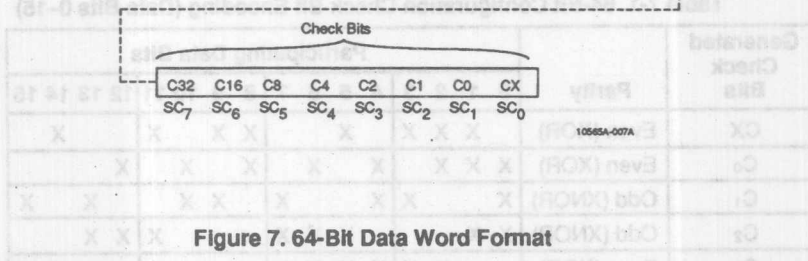


Figure 7. 64-Bit Data Word Format

Chip Configuration

Two Am29C660 EDC Circuits connected as shown in Figure 8, provide all the necessary logic for all single-bit error detection correction, all double-bit error detection,

and some triple-bit error detection on a 64-bit data field. In this configuration, eight check bits are required. CODE ID₀, 1 control signal inputs distinguish the upper 32 bits from the lower 32 bits as shown in Table 1.

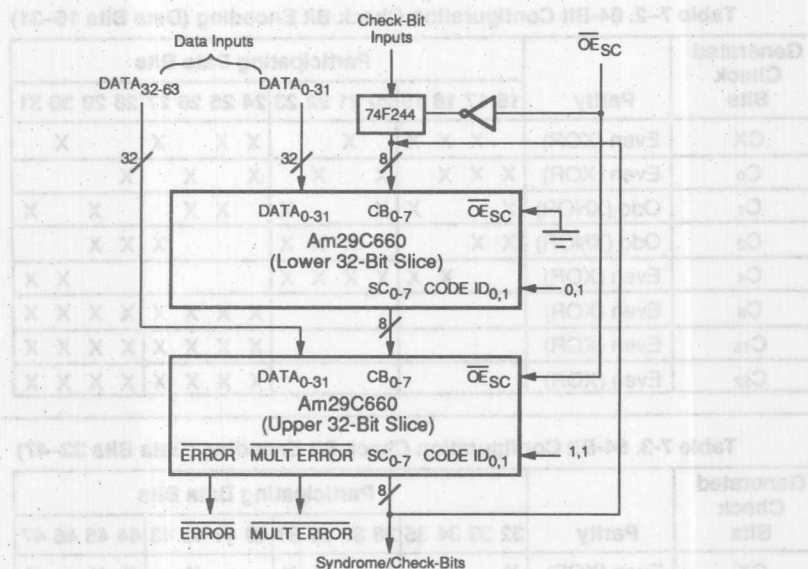


Figure 8. 64-Bit Configuration

The valid syndrome bits and the $\overline{\text{ERROR}}$ and $\overline{\text{MULT}}$ $\overline{\text{ERROR}}$ output signals are taken from the upper EDC slice. The lower EDC slice has its $\overline{\text{OE}}_{\text{Esc}}$ input tied to ground (logic "0") and its SC_{0-7} outputs connected to the respective CB_{0-7} inputs of the upper EDC slice. The 32 most significant data bits, DATA_{32-63} , are connected to the data lines of the upper EDC slice. All the latch enables and control signals must be input to both the upper and the lower EDC slices.

In the Detect/Correct Mode, the valid syndrome bits from the upper EDC slice must be read into the lower EDC slice via the check bit inputs of the lower EDC slice. They are selected by an internal MUX as inputs to the bit-in-error decoder (see Block Diagram). External buffering and output enabling of the check bit lines is required as shown in Figure 8. The $\overline{\text{OE}}_{\text{Esc}}$ signal to the upper EDC slice can be used to control the enabling of the check bits to the lower EDC slice. The external check bits to the lower EDC slice are disabled.

Generate Mode

In this mode, check bits will be generated that correspond to the contents of the Data Input Latch. The generated check bits are placed on the SC₀₋₇ outputs of the upper EDC slice.

Check bits are generated according to a modified Hamming Code. Details of the code for check bit generation are shown in Tables 7-1 through 7-4. Check bits are generated as either an XOR or XNOR of 32 of the 64 bits. The XOR function results in an even parity check bit, the XNOR in an odd parity check bit.

Table 7-1. 64-Bit Configuration Check Bit Encoding (Data Bits 0–15)

Generated Check Bits	Parity	Participating Data Bits															
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CX	Even (XOR)		X	X	X		X			X	X	X				X	
C ₀	Even (XOR)	X	X	X		X		X		X		X		X			
C ₁	Odd (XNOR)	X			X	X			X		X	X			X		X
C ₂	Odd (XNOR)	X	X				X	X	X			X		X	X	X	
C ₄	Even (XOR)			X	X	X	X	X	X							X	X
C ₈	Even (XOR)									X	X	X	X	X	X	X	X
C ₁₆	Even (XOR)	X	X	X	X	X	X	X	X								
C ₃₂	Even (XOR)	X	X	X	X	X	X	X	X								

Table 7-2. 64-Bit Configuration Check Bit Encoding (Data Bits 16–31)

Generated Check Bits	Parity	Participating Data Bits															
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
CX	Even (XOR)		X	X	X		X			X	X	X				X	
C ₀	Even (XOR)	X	X	X		X		X		X		X		X			
C ₁	Odd (XNOR)	X			X	X			X		X	X			X		X
C ₂	Odd (XNOR)	X	X				X	X	X				X	X	X		
C ₄	Even (XOR)			X	X	X	X	X	X							X	X
C ₈	Even (XOR)									X	X	X	X	X	X	X	X
C ₁₆	Even (XOR)									X	X	X	X	X	X	X	X
C ₃₂	Even (XOR)									X	X	X	X	X	X	X	X

Table 7-3. 64-Bit Configuration Check Bit Encoding (Data Bits 32–47)

Generated Check Bits	Parity	Participating Data Bits															
		32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
CX	Even (XOR)	X				X		X	X			X		X	X	X	
C ₀	Even (XOR)	X	X	X		X		X		X		X		X			
C ₁	Odd (XNOR)	X			X	X			X		X	X			X		X
C ₂	Odd (XNOR)	X	X				X	X	X				X	X	X		
C ₄	Even (XOR)			X	X	X	X	X	X							X	X
C ₈	Even (XOR)									X	X	X	X	X	X	X	X
C ₁₆	Even (XOR)	X	X	X	X	X	X	X	X								
C ₃₂	Even (XOR)									X	X	X	X	X	X	X	X

Table 7-4. 64-Bit Configuration Check Bit Encoding (Data Bits 48–63)

Generated Check Bits	Parity	Participating Data Bits															
		48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
CX	Even (XOR)	X				X	X	X			X			X	X		X
C ₀	Even (XOR)	X	X	X		X		X		X		X		X			
C ₁	Odd (XNOR)	X			X	X		X		X	X			X		X	
C ₂	Odd (XNOR)	X	X			X	X	X			X		X	X	X		
C ₄	Even (XOR)			X	X	X	X	X	X							X	X
C ₈	Even (XOR)									X	X	X	X	X	X	X	X
C ₁₆	Even (XOR)									X	X	X	X	X	X	X	X
C ₃₂	Even (XOR)	X	X	X	X	X	X	X	X								

The check bit is generated as either an XOR or XNOR of the 32 data bits noted by an "X" in the table.

Detect Mode

In this mode, the device will compare the check bits generated from the contents of the Data Input Latch with the contents of the Check Bit Latch. All single- and double-bit errors and some triple-bit errors will be detected. If one or more errors are detected, ERROR goes LOW. If two or more errors are detected, both ERROR and MULT ERROR will go LOW. Both error indicators will be HIGH if no errors are detected. The valid ERROR and MULT ERROR signals are from the upper EDC slice.

The syndrome bits which are generated during error detection are available on the outputs SC₀₋₇ of the upper EDC slice. The syndrome bits may be decoded to determine if a bit error was detected, and for a single-bit error, which of the data or check bits is in error. Table 8 shows the syndrome bit decoding for the 64-bit data word configuration. If no error is detected, the syndrome bits will all be zeros.

In the Detect Mode, the contents of the Data Input Latch are driven directly into the inputs of the Data Output Latch without correction.

Table 8. 64-Bit Syndrome Bit Decoding Matrix

Syndrome Bits				S32	S16	S8	S4														
SX	S0	S1	S2	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
0	0	0	0	*	C32	C16	T	C8	T	T	M	C4	T	T	M	T	46	62	T		
0	0	0	1	C2	T	T	M	T	43	59	T	T	53	37	T	M	T	T	M		
0	0	1	0	C1	T	T	M	T	41	57	T	T	51	35	T	15	T	T	31		
0	0	1	1	T	M	M	T	13	T	T	29	23	T	T	7	T	M	M	T		
0	1	0	0	C0	T	T	M	T	40	56	T	T	50	34	T	M	T	T	M		
0	1	0	1	T	49	33	T	12	T	T	28	22	T	T	6	T	M	M	T		
0	1	1	0	T	M	M	T	10	T	T	26	20	T	T	4	T	M	M	T		
0	1	1	1	16	T	T	0	T	M	M	T	T	M	M	T	M	T	T	M		
1	0	0	0	CX	T	T	M	T	M	M	T	T	M	M	T	14	T	T	30		
1	0	0	1	T	M	M	T	11	T	T	27	21	T	T	5	T	M	M	T		
1	0	1	0	T	M	M	T	9	T	T	25	19	T	T	3	T	47	63	T		
1	0	1	1	M	T	T	M	T	45	61	T	T	55	39	T	M	T	T	M		
1	1	0	0	T	M	M	T	8	T	T	24	18	T	T	2	T	M	M	T		
1	1	0	1	17	T	T	1	T	44	60	T	T	54	38	T	M	T	T	M		
1	1	1	0	M	T	T	M	T	42	58	T	T	52	36	T	M	T	T	M		
1	1	1	1	T	48	32	T	M	T	T	M	M	T	T	M	T	M	M	T		

* = No errors detected

Number = The bit number of the single bit-in-error

T = Two errors detected

M = More than two errors detected

Detect/Correct Mode

In this mode, the EDC functions the same way as in the Detect Mode except that the correction network is enabled to correct, by complementing, any single-bit error in the Data Input Latch before placing the data on the inputs of the Data Output Latch. If a multiple error is detected, the output of the correction network is undefined. If a single-bit error occurs to a check bit, there is no automatic correction. If a check bit correction is desired, it can be done by placing the device in the Generate Mode to produce the correct check bit sequence for the data in the Data Input Latch.

For data correction, both the upper EDC slice and the lower EDC slice require access to the syndrome bits SC₀₋₇ of the upper EDC slice. The EDC slice has access to these syndrome bits through an internal data path. The syndrome bits must be read through the CB₀₋₇ inputs for the lower EDC slice as shown in Figure 8. In the Detect/Correct Mode, the valid SC₀₋₇ outputs of the up-

per EDC slice must be enabled so that they are available for reading in through the CB₀₋₇ inputs of the lower EDC slice.

Pass-Thru Mode

In this mode, the unmodified contents of the Data Input Latch are placed on the inputs of the Data Output Latch, and the contents of the Check bit Latch are placed on the output SC₀₋₆ (SC₇ is a logical "1"). ERROR and MULT ERROR are forced HIGH in this mode (inactive).

Diagnostic Generate Mode

This is one of the three special Diagnostic Modes selected by the control inputs DIAG MODE_{0,1}. This mode is similar to the normal Generate Mode except that the check bits are not generated from the contents of the Data Input Latch. They are instead taken directly from the contents of the Diagnostic Latch. Table 9 shows the Diagnostic Latch coding format.

Table 9. 64-Bit Diagnostic Latch Coding Format

Data Bit	Internal Function
0	Diagnostic Check Bit X
1	Diagnostic Check Bit 0
2	Diagnostic Check Bit 1
3	Diagnostic Check Bit 2
4	Diagnostic Check Bit 4
5	Diagnostic Check Bit 8
6	Diagnostic Check Bit 16
7	Diagnostic Check Bit 32
8	CODE ID ₀ , Lower 32-Bit Slice
9	CODE ID ₁ , Lower 32-Bit Slice
10	DIAG MODE ₀ , Lower 32-Bit Slice
11	DIAG MODE ₁ , Lower 32-Bit Slice
12	CORRECT, Lower 32-Bit Slice
13-31	Don't Care
32-39	Don't Care
40	CODE ID ₀ , Upper 32-Bit Slice
41	CODE ID ₁ , Upper 32-Bit Slice
42	DIAG MODE ₀ , Upper 32-Bit Slice
43	DIAG MODE ₁ , Upper 32-Bit Slice
44	CORRECT, Upper 32-Bit Slice
45-63	Don't Care

Diagnostic Detect, Diagnostic Detect /Correct Modes

These two special Diagnostic Modes are also selected by the control inputs DIAG MODE_{0,1}. These modes are similar to the normal Detect and Detect/Correct Modes

except that the check bits are taken from the Diagnostic Latch, rather than from the Check Bit Input Latch.

Initialize Mode

In this mode the inputs of the Data Output Latch are forced to all zeros. The syndrome bit outputs SC_{0-6} are generated to correspond to the all-zero data on the Data Output Latch. SC_7 is tied high. The \overline{ERROR} and $\overline{MULT\ ERROR}$ outputs are forced HIGH in this mode. The Initialize Mode is useful after power-up when RAM contents are random. The EDC may be placed in this mode

and its outputs written into all memory addresses under processor control.

Internal Control Mode

This mode is selected by the external control inputs $CODE\ ID_{0-1}$. When in the Internal Control Mode, the Am29C660 takes the $CODE\ ID_{0-1}$, $DIAG\ MODE_{0-1}$, and $CORRECT$ control signals from the Diagnostic Latch rather than from the external input lines.

Table 10. Key AC Calculations For The 64-Bit Configuration

64-Bit Propagation Delay		Component Delays from Am29C660 AC Specifications	Example for Am29C660E
From	To		
$DATA_{0-31}$	Check Bit Outputs	(Data to SC, $CODE\ ID\ 10$) + (Check Bit to SC, $CODE\ ID\ 11$)	$11 + 9 = 20\ ns$
$DATA_{0-31}$	Corrected Data Outputs	(Data to SC, + $CODE\ ID\ 10$) + (Check Bit to SC, $CODE\ ID\ 11$) + (Check Bit to Data, $CODE\ ID\ 10$)	$11 + 9 + 10 = 30\ ns$
$DATA_{0-31}$	Syndrome Outputs	(Data to SC, $CODE\ ID\ 10$) + (Check Bit to SC, $CODE\ ID\ 11$)	$11 + 9 = 20\ ns$
$DATA_{0-31}$	\overline{ERROR} Outputs	(Data to SC, $CODE\ ID\ 10$) + (Check Bit to \overline{ERROR} , $CODE\ ID\ 11$)	$11 + 7 = 18\ ns$
$DATA_{0-31}$	$\overline{MULT\ ERROR}$ Outputs	(Data to SC, $CODE\ ID\ 10$) + (Check Bit to $\overline{MULT\ ERROR}$, $CODE\ ID\ 11$)	$11 + 9 = 20\ ns$

APPLICATIONS

System Design Considerations

To obtain optimum performance and maximum design flexibility, AMD's Dynamic Memory Management System has been divided into functional building blocks. For 32-bit error detecting/correcting systems, these building blocks include the Am29C660 EDC Circuit, Am29C668 4M Configurable Dynamic Memory Controller/Driver, and the Am2971A 100-MHz Enhanced Programmable Event Generator or delay lines as the timing reference. Together these chips can perform traditional EDC, Flow-Thru or Fly-By, or AMD's Scrubbing EDC cycle.

High-Performance Parallel Operation

For maximum memory system performance, the EDC should be used in the Check-Only (or Fly-By) configuration shown in Figure 9. With this configuration, the memory system operates as fast with EDC as it would without.

On reads from memory, data is read out from the DRAMs directly to the data bus (same as in a non-EDC system). At the same time, the data is read into the EDC to check for errors. If an error exists, the EDC's error flags (\overline{ERROR} , $\overline{MULT\ ERROR}$) are used to interrupt the CPU and/or to stretch the memory cycle. If no error is detected, no slowdown is required.

If an error is detected, the EDC generates corrected data for the processor. At the designer's option, the correct data may be written back into memory; error logging

and diagnostic routines may also be run under processor control.

The Check-Only configuration allows data reads to proceed as fast with EDC as without. Only if an error is detected is there any slowdown, but even if the memory system had an error every hour this would mean only one error every 3 – 4 billion memory cycles. Therefore, even with a very high error rate, EDC in a Check-Only configuration has essentially zero impact on memory system speed.

On writes to memory, check bits must be generated before the full memory word can be written into memory. Using the Am29C983 Multiple Bus Exchange allows the data word to be buffered on the memory board while check bits are generated. This makes the check bit generate time transparent to the processor.

EDC in the Data Path

The simplest configuration for the Am29C660 is to have the EDC circuit directly in the data path, as shown in Figure 10 (Flow-Thru configuration). In this configuration, data read from memory is always corrected prior to putting the data on the data bus. The advantages are simpler operation and no need for mid-cycle interrupts. The disadvantage is that memory system speed is slowed by the amount of time it takes for error correction on every cycle.

Usually the Flow-Thru configuration will be used with MOS microprocessors which have ample memory timing budgets. Most high-performance processors will use the high-performance parallel configuration shown in Figure 9 (Fly-By).

Memory Scrubbing During Refresh

Scrubbing is an error correction technique that examines the entire memory during system refresh cycles, thus causing little or no impact on performance yet providing high data reliability since errors cannot accumulate.

Single-bit errors are by far the most common in a dynamic memory system, and are always correctable by the EDC. Double-bit errors occur far less frequently than single-bit errors (100-to-1 or greater) and are always detected by the EDC, but not corrected.

In a memory system, soft errors occur only one at a time. A double-bit error in a data word occurs when a single soft error is left uncorrected and is followed by another error in the data word within hours, days, or weeks after the first occurrence.

Scrubbing the memory periodically avoids almost all double-bit errors. In the scrubbing operation, every data word in memory is periodically checked by the EDC for single-bit errors. If one is found, it is corrected and the correct data word is written back into memory. Errors are not allowed to accumulate, and thus most double-bit errors are avoided.

The scrubbing operation is generally done as a background routine when the memory is not being used by the processor. If memory is scrubbed frequently, errors that are detected and corrected during processor accesses need not be immediately written back into memory. Instead, the error will be corrected in memory during scrubbing. This reduces the time delay involved in a processor access of an incorrect memory word.

On each refresh-with-scrub cycle, one memory location is read, checked for errors, and if necessary, corrected before being written back into memory. For a sixteen-megaword memory (2^{24} locations) with one refresh every 16 microseconds, the AMD Memory Management chip set will scrub the entire memory of single-bit errors every minute. If multiple-bit errors are encountered during a scrub cycle, \overline{WE} is suppressed.

With the occurrence of an error, a read/modify/write (R/M/W) cycle is performed. The duration of a R/M/W cycle is typically longer than a normal read or write cycle. During refresh operations, a row in each bank is accessed

by energizing the \overline{RAS} line. This refreshes all the locations in that row. If an error is detected, a write operation is performed in that bank at the location of the error. This is accomplished by energizing the \overline{CAS} line in that bank for that location. The entire checking operation is performed within the refresh cycle. A wait state may need to be issued to extend the cycle should an error be discovered. However, the system reliability will be increased because soft errors will not be able to accumulate in areas of memory that are not frequently accessed.

When performing refresh without scrubbing, all four \overline{RAS} lines go LOW but the \overline{CAS} lines remain HIGH or inactive. A refresh with scrubbing will activate all four \overline{RAS} lines as before and a single \overline{CAS} line. Errors that are detected during scrubbing cycles do not cause interrupts or bus-error (BERR) signal assertions.

Figure 11 shows a sixteen-megaword memory system with error detection and correction.

Check Bit Correction

The EDC detects single bit errors whether the error is a data bit or a check bit. Data bit errors are automatically corrected by the EDC. To generate corrected check bits once a single check bit error is detected, the EDC need only be switched to GENERATE mode (data in the DATA INPUT LATCH is valid).

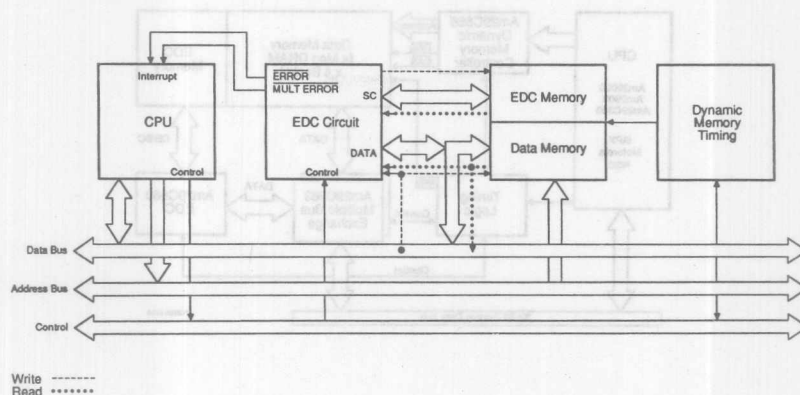
The syndromes generated by the EDC may be decoded to determine whether the single bit error is a check bit.

In many memory systems, a check bit error will be ignored on the memory read and corrected during a periodic "scrubbing" of memory (see section in System Design Considerations).

Multiple Errors

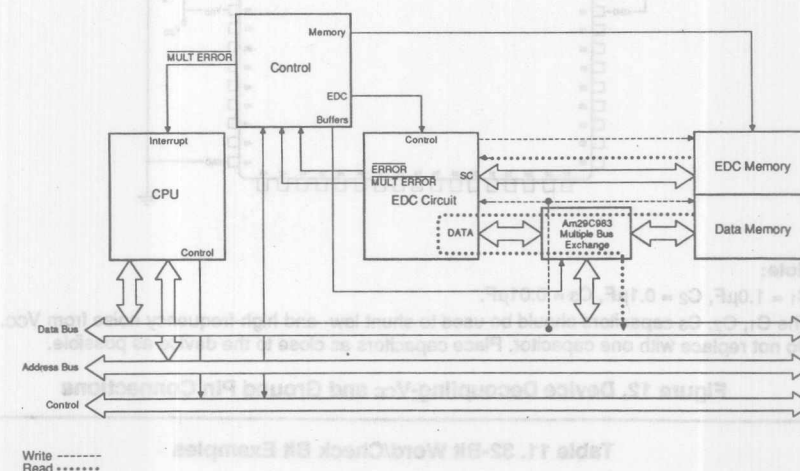
The bit-in-error decode logic uses syndrome bit S0 through S32 to correct errors, SX is only used in developing the multiple error signal. This means that some multiple errors will cause a data bit to be inverted.

For example, in the 16-bit mode if data bits 8 and 13 are in error the syndrome 111100 (SX, S0, S1, S4, S8) is produced. This is flagged a double error by the error detection logic, but the bit-in-error decoder only receives syndrome 11100 (S0, S1, S2, S4, S8), which it decodes as a single error in data bit 0 and inverts that bit. If it is desired to inhibit this version the multiple error output may be connected to the correct input as in Figure 13. This will inhibit correction when a multiple error occurs. Extra time delay may be introduced in the data to correct data path when this is done.



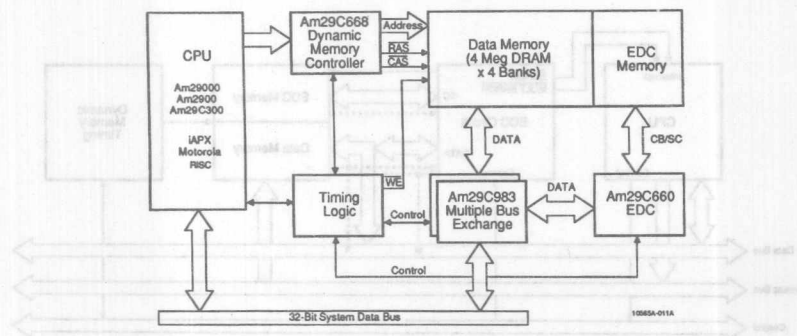
10565-009A

Figure 9. Check-Only Configuration



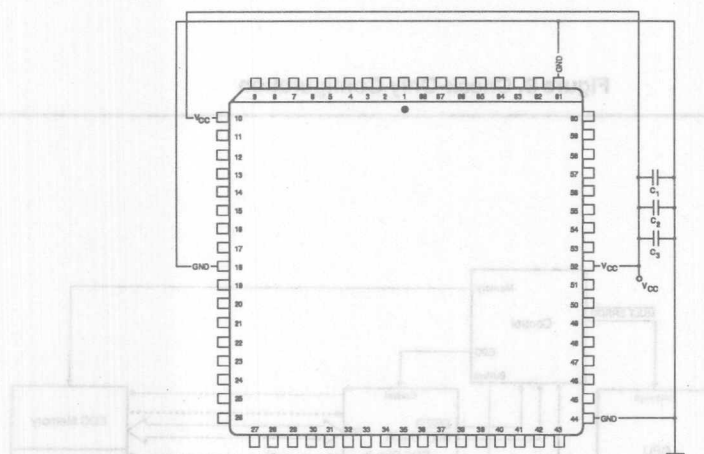
10565-010A

Figure 10. Correct-Always Configuration



10565-011A

Figure 11. Sixteen-Megaword Memory System with EDC



10565-012A

Note:

$C_1 = 1.0\mu\text{F}$, $C_2 = 0.1\mu\text{F}$, $C_3 = 0.01\mu\text{F}$.

The C_1 , C_2 , C_3 capacitors should be used to shunt low- and high-frequency noise from V_{cc} . Do not replace with one capacitor. Place capacitors as close to the device as possible.

Figure 12. Device Decoupling- V_{cc} and Ground Pin Connections

Table 11. 32-Bit Word/Check Bit Examples

Example 32-Bit Word								Corresponding Check Bits	
D_{31}							D_0	C_X	C_{16}
0101	0101	0101	0101	0101	0101	0101	0101	0011000	
1010	1010	1010	1010	1010	1010	1010	1010	0011000	
0001	0000	1100	0111	0111	1101	0111	1111	1101110	
0000	0011	0011	1101	1000	0101	0100	0000	1110011	
1111	1111	1111	0000	0000	0000	1111	1110	0101001	

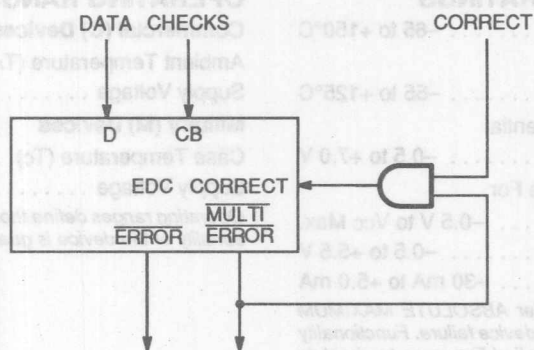


Figure 13. Inhibition of Data Modification

10565-013A

ABSOLUTE MAXIMUM RATINGS

Storage Temperature	-65 to +150°C
Temperature (Case)	
Under Bias	-55 to +125°C
Supply Voltage to Ground Potential	
Continuous	-0.5 to +7.0 V
DC Voltage Applied to Outputs For	
HIGH Output State	-0.5 V to V_{CC} Max.
DC Input Voltage	-0.5 to +5.5 V
DC Input Current	-30 mA to +5.0 mA

Stresses above those listed under ABSOLUTE MAXIMUM RATINGS may cause permanent device failure. Functionality at or above these limits is not implied. Exposure to absolute maximum ratings for extended periods may affect device reliability.

OPERATING RANGES**Commercial (C) Devices**

Ambient Temperature (T_A)	0 to +70°C
Supply Voltage	+4.75 to +5.25 V

Military (M) Devices

Case Temperature (T_c)	-55 to +125°C
Supply Voltage	+4.5 to +5.5 V

Operating ranges define those limits between which the functionality of the device is guaranteed.

DC CHARACTERISTICS over operating range unless otherwise specified (for APL Products, Group A, Subgroups 1, 2, 3 are tested unless otherwise noted)

Parameter Symbol	Parameter Description	Test Conditions (Note 1)	Min.	Max.	Unit
V_{IH}	Input HIGH Voltage	Guaranteed Input Logical HIGH Voltage for all Inputs (Note 3)	2.0		V
V_{IL}	Input LOW Voltage	Guaranteed Input Logical LOW Voltage for all Inputs (Note 3)		0.8	V
I_{IH}	Input HIGH Current	$V_{CC} = \text{Max.}$, $V_{IN} = V_{CC}$		5.0	μA
I_{IL}	Input LOW Current	$V_{CC} = \text{Max.}$, $V_{IN} = \text{GND}$		-5.0	μA
V_{OH}	Output HIGH Voltage	$V_{CC} = \text{Min.}$	$I_{OH} = -300 \mu\text{A}$	$V_{CC} - 0.2$	V
			MIL $I_{OH} = -8 \text{ mA}$	2.4	
			COM'L $I_{OH} = -15 \text{ mA}$	2.4	
V_{OL}	Output LOW Voltage	$V_{CC} = \text{Min.}$	$I_{OL} = 300 \mu\text{A}$	0.2	V
			MIL $I_{OL} = 8 \text{ mA}$	0.5	
			COM'L $I_{OL} = 15 \text{ mA}$	0.5	
I_{OZ}	Off-State (High-Impedance) Output Current	$V_{CC} = \text{Max.}$	$V_O = 0 \text{ V}$ $V_O = V_{CC} (\text{Max.})$	-10 10	μA
I_{OS}	Output Short-Circuit Current	$V_{CC} = \text{Min.}$, $V_O = 0 \text{ V}$ (Note 2)		-30	mA
I_{CCQ}	Quiescent Power Supply Current (CMOS Inputs)	$V_{CC} = \text{Max.}$, $V_{CC} - 0.2 \text{ V} \leq V_{IN}, V_{IN} \leq 0.2 \text{ V}$ $f_{OP} = 0$		5.0	mA
I_{CCT}	Quiescent Input Power Supply Current (per Input @ TTL HIGH) (Note 4)	$V_{CC} = \text{Max.}$, $V_{IN} = 3.4 \text{ V}$, $f_{OP} = 0$		0.5	mA/ Input
I_{CCD}	Dynamic Power Supply Current	$V_{CC} = \text{Max.}$, $V_{CC} - 0.2 \text{ V} \leq V_{IN}, V_{IN} \leq 0.2 \text{ V}$ Outputs Open, $OE = \text{LOW}$	MIL	6.5	mA/ MHz
			COM'L	6	
I_{CC}	Total Power Supply Current (Note 5)	$V_{CC} = \text{Max.}$, $f_{OP} = 10 \text{ MHz}$ Outputs open, $OE = \text{LOW}$ 50% Duty cycle $V_{CC} - 0.2 \text{ V} \leq V_{IN}, V_{IN} \leq 0.2 \text{ V}$	MIL	70	mA
			COM'L	65	
		$V_{CC} = \text{Max.}$, $f_{OP} = 10 \text{ MHz}$ Outputs open, $OE = \text{LOW}$ 50% Duty cycle $V_{IH} = 3.4 \text{ V}$, $V_{IL} = 0.4 \text{ V}$	MIL	70	
			COM'L	65	

- Notes: 1. For conditions shown as Min. or Max., use appropriate value specified under Operating Range for the applicable device type.
2. Not more than one output should be shorted at a time. Duration of the short-circuit test should not exceed one second.
3. These input levels provide zero noise immunity and should only be static tested in a noise-free environment.
4. I_{CCT} is derived by measuring the total current with all the inputs tied together at 3.4 V, subtracting out I_{CCQ} , then dividing by the total number of inputs.
5. Total Power Supply Current is the sum of the Quiescent Current and the Dynamic Current (at either CMOS or TTL input levels). For all conditions, the Total Power Supply Current can be calculated by using the following equation:

$$I_{CC} = I_{CCQ} + I_{CCT} (N_T \times D_H) + I_{CCD} (f_{OP})$$

$$D_H = \text{Data duty cycle TTL HIGH period } (V_{IN} = 3.4 \text{ V}).$$

$$N_T = \text{Number of dynamic inputs driven at TTL levels.}$$

$$f_{OP} = \text{Operating frequency in Megahertz.}$$

CAPACITANCE (Note 1)

Parameter Symbol	Parameter Description	Test Conditions	Typical	Unit
$C_{IN} (\text{CB})$	Input Capacitance (CB)	$T_A = 25^\circ\text{C}$	5	pF
$C_{OUT} (\text{SC})$	Output Capacitance (SC)	$f = 1 \text{ MHz}$	6	pF
$C_{VO} (\text{Data})$	I/O Capacitance		6	pF

Note:

1. These parameters are not tested in production and are not guaranteed, but are evaluated at initial characterization and at any time the design is modified where capacitance may be effected.

SWITCHING CHARACTERISTICS over COMMERCIAL operating ranges (Notes 1 and 2)

The following table specifies the guaranteed device performance over the Commercial operating range of 0°C to +70°C (ambient), with V_{CC} 4.75 to 5.25 V. All input switching is between 0 V and 3 V at 1 V/ns, and measurements are made at 1.5 V. All outputs have maximum DC load. All units are in nanoseconds (ns).

No.	Parameter Symbol	Data Path Description		Am29C660		Am29C660A		Am29C660B		Am29C660C	
		From Input	To Output	Min.	Max.	Min.	Max.	Min.	Max.	Min.	Max.
1	A	tPD	DATA ₀₋₃₁ (Note 3)	SC ₀₋₇	37		27		25		18
	B			DATA ₀₋₃₁ (Note 2)	49		36		30		24
	C			ERROR	40		30		25		16
	D			MULT ERROR	45		33		27		20
2	A	tPD	CB ₀₋₇ (CODE ID 00, 11)	SC ₀₋₇	22		16		14		14
	B			DATA ₀₋₃₁	46		34		30		21
	C			ERROR	26		19		17		13
	D			MULT ERROR	31		23		20		16
3	A	tPD	CB ₀₋₇ (CODE ID 10)	SC ₀₋₇	22		16		16		15
	B			DATA ₀₋₃₁	30		20		18		16
4	A	tPD	GENERATE	SC ₀₋₇	29		21		21		18
	B			ERROR	30		25		23		9
	C			MULT ERROR	30		25		23		11
5		tPD	CORRECT (Not Internal Control Mode)	DATA ₀₋₃₁	31		23		23		16
6	A	tPD	DIAG MODE _{0,1} (Not Internal Control Mode)	SC ₀₋₇	23		17		17		16
	B			DATA ₀₋₃₁	35		26		26		26
	C			ERROR	27		20		20		11
	D			MULT ERROR	33		24		24		20
7	A	tPD	CODE ID _{0,1}	SC ₀₋₇	25		18		18		18
	B			DATA ₀₋₃₁	35		26		26		23
	C			ERROR	29		21		21		17
	D			MULT ERROR	35		26		26		21
8	A	tPD	LE _{IN} (From Latched to Transparent)	SC ₀₋₇	37		27		27		22
	B			DATA ₀₋₃₁	51		38		38		28
	C			ERROR	41		30		30		19
	D			MULT ERROR	45		33		33		22
9		tPD	LE _{OUT} (From Latched to Transparent)	DATA ₀₋₃₁	17		12		12		12
10	A	tPD	LE _{DIAG} (From Latched to Transparent; Not Internal Control Mode)	SC ₀₋₇	21		15		15		15
	B			DATA ₀₋₃₁	38		29		29		24
	C			ERROR	26		19		19		15
	D			MULT ERROR	30		22		22		19

SWITCHING CHARACTERISTICS over COMMERCIAL operating range (Continued)

No.		Parameter Symbol	Data Path Description		Am29C660		Am29C660A		Am29C660B		Am29C660C		
			From Input	To Output	Min.	Max.	Min.	Max.	Min.	Max.	Min.	Max.	
11	A	tPD	Internal Control Mode: LE DIAG (From Latched to Transparent)	SC0-7		22		16		16		16	
	B			DATA0-31		42		32		32		22	
	C			ERROR		26		19		19		16	
	D			MULT ERROR		33		24		24		18	
12	A	tPD	Internal Control Mode: DATA0-31 (Via Diagnostic Latch)	SC0-7		22		16		16		16	
	B			DATA0-31 (Note 2)		42		32		32		25	
	C			ERROR		27		20		20		13	
	D			MULT ERROR		34		25		25		16	
13		tSET	DATA0-31 (Note 4)	LEIN	6		5		4		3		
14		tHOLD			4		4		4		4		
15		tSET	CB0-7 (Note 4)		5		5		4		2		
16		tHOLD			4		4		4		4		
17		tSET	DATA0-31 (Note 4)	LEOUT	30		23		19		6		
18		tHOLD			0		0		0		0		
19		tSET	CB0-7 (Note 4) (CODE ID 00, 11)		20		15		15		8		
20		tHOLD			0		0		0		0		
21		tSET	CB0-7 (Note 4) (CODE ID 10)		20		15		15		14		
22		tHOLD			0		0		0		0		
23		tSET	CORRECT (Note 4)		16		11		11		8		
24		tHOLD			0		0		0		0		
25		tSET	DIAG MODE0, 1 (Note 4)		23		17		17		17		
26		tHOLD			0		0		0		0		
27		tSET	CODE ID0, 1 (Note 4)		23		17		17		10		
28		tHOLD			0		0		0		0		
29		tSET	LEIN (Note 4)		31		25		20		19		
30		tHOLD			0		0		0		0		
31		tSET	DATA0-31 (Note 4)		LEDIAG	6		5		4		3	
32		tHOLD			3		3		3		3		
33		tEN	OE BYTE0-3 (Note 5)	DATA0-31		27		23		23		8	
34		tDIS				23		19		19		11	
35		tEN	OE SC (Note 5)	SC0-7		28		24		24		17	
36		tDIS				24		20		20		13	
37		tpw	Minimum Pulse Width: LEIN, LEOUT, LEDIAG		12		9		9		6		

Notes:

1. $C_L = 50$ pF.
2. Certain parameters are combinational propagation delay calculations.
3. Data In or LEIN to Correct Data Out measurement requires timing as shown in the Switching Waveforms.
4. Setup and Hold times relative to Latch Enables (Latching up data).
5. Output disable tests specified with $C_L = 5$ pF and measured to 0.5 V change of output voltage level. Testing is performed at $C_L = 50$ pF and correlated to $C_L = 5$ pF.

SWITCHING CHARACTERISTICS over COMMERCIAL operating ranges (Notes 1 and 2)

No.		Parameter Symbol	Data Path Description		Am29C660D		PRELIMINARY		Unit
					Am29C660E		Min.	Max.	
			From Input	To Output	Min.	Max.			
1	A	tpD	DATA0-31 (Note 3)	SC0-7		14		11	ns
	B			DATA0-31 (Note 2)		18		14	ns
	C			ERROR		12		9	ns
	D			MULT ERROR		15		12	ns
2	A	tpD	CB0-7 (CODE ID 00, 11)	SC0-7		11		9	ns
	B			DATA0-31		16		12	ns
	C			ERROR		10		7	ns
	D			MULT ERROR		12		9	ns
3	A	tpD	CB0-7 (CODE ID 10)	SC0-7		12		9	ns
	B			DATA0-31		12		10	ns
4	A	tpD	GENERATE	SC0-7		14		12	ns
	B			ERROR		7		6	ns
	C			MULT ERROR		8		7	ns
5		tpD	CORRECT (Not Internal Control Mode)	DATA0-31		12		10	ns
6	A	tpD	DIAG MODE0, 1 (Not Internal Control Mode)	SC0-7		12		11	ns
	B			DATA0-31		20		15	ns
	C			ERROR		10		8	ns
	D			MULT ERROR		15		14	ns
7	A	tpD	CODE ID0, 1	SC0-7		14		12	ns
	B			DATA0-31		18		14	ns
	C			ERROR		13		12	ns
	D			MULT ERROR		16		15	ns
8	A	tpD	LEIN (From Latched to Transparent)	SC0-7		17		15	ns
	B			DATA0-31 (Note 2)		21		17	ns
	C			ERROR		14		12	ns
	D			MULT ERROR		17		13	ns
9		tpD	LEOUT (From Latched to Transparent)	DATA0-31		9		8	ns
10	A	tpD	LEDIAG (From Latched to Transparent; Not Internal Control Mode)	SC0-7		12		11	ns
	B			DATA0-31		18		14	ns
	C			ERROR		12		11	ns
	D			MULT ERROR		14		12	ns

Notes:
1. CL = 50 pF.
2. Certain parameters are conditional propagation delay calculations.
3. Data in or LE_{IN} to Correct Data Out measurement requires timing as shown in the Switching Waveforms.
4. Setup and Hold times relative to Latch Enable (latching up test).
5. Output disable tests specified with CL = 5 pF and measured to 0.5 V change of output voltage level. Testing is performed at CL = 50 pF and correlated to CL = 5 pF.

SWITCHING CHARACTERISTICS over COMMERCIAL operating range (Continued)

No.		Parameter Symbol	Data Path Description		Am29C660D		PRELIMINARY Am29C660E		Unit
					Min.	Max.	Min.	Max.	
			From Input	To Output					
11	A	tPD	Internal Control Mode: LE DIAG (From Latched to Transparent)	SC0-7		12		11	ns
	B			DATA0-31		17		14	ns
	C			ERROR		12		10	ns
	D			MULT ERROR		14		12	ns
12	A	tPD	Internal Control Mode: DATA0-31 (Via Diagnostic Latch)	SC0-7		12		10	ns
	B			DATA0-31 (Note 2)		19		15	ns
	C			ERROR		10		9	ns
	D			MULT ERROR		12		11	ns
13	tSET	DATA0-31 (Note 4)	LEIN	3		3		ns	
14	tHOLD			3		3		ns	
15	tSET	CB0-7 (Note 4)		2		2		ns	
16	tHOLD			3		3		ns	
17	tSET	DATA0-31 (Note 4)	LEOUT	5		5		ns	
18	tHOLD			0		0		ns	
19	tSET	CB0-7 (Note 4) (CODE ID 00, 11)		11		11		ns	
20	tHOLD			0		0		ns	
21	tSET	CB0-7 (Note 4) (CODE ID 10)		6		6		ns	
22	tHOLD			0		0		ns	
23	tSET	CORRECT (Note 4)		6		6		ns	
24	tHOLD			0		0		ns	
25	tSET	DIAG MODE0, 1 (Note 4)		13		13		ns	
26	tHOLD			0		0		ns	
27	tSET	CODE ID0, 1 (Note 4)		8		8		ns	
28	tHOLD			0		0		ns	
29	tSET	LEIN (Note 4)		14		14		ns	
30	tHOLD			0		0		ns	
31	tSET	DATA0-31 (Note 4)	LEDIAG	3		3		ns	
32	tHOLD		3		3		ns		
33	tEN	OE BYTE0-3 (Note 5)	DATA0-31		7	2	5	ns	
34	tDIS				8	2	8	ns	
35	tEN	OE sc (Note 5)	SC0-7		8	2	5	ns	
36	tDIS				10	2	8	ns	
37	tpw	Minimum Pulse Width: LEIN, LEOUT, LEDIAG			5		5		ns

Notes:

1. $C_L = 50$ pF.
2. Certain parameters are combinational propagation delay calculations.
3. Data In or LEIN to Correct Data Out measurement requires timing as shown in the Switching Waveforms.
4. Setup and Hold times relative to Latch Enables (Latching up data).
5. Output disable tests specified with $C_L = 5$ pF and measured to 0.5 V change of output voltage level. Testing is performed at $C_L = 50$ pF and correlated to $C_L = 5$ pF.

SWITCHING CHARACTERISTICS over MILITARY operating range (for APL Products, Group A, Subgroups 9, 10, 11 are tested unless otherwise noted) (Notes 1 and 2)

The following table specifies the guaranteed device performance over the Military and Extended-Commercial operating ranges of -55°C to $+125^{\circ}\text{C}$ (case), with V_{CC} 4.5 to 5.5 V and 4.75 to 5.25 V, respectively. All input

switching is between 0 V and 3 V at 1 V/ns and measurements are made at 1.5 V. All outputs have maximum DC load. All units are in nanoseconds (ns).

No.		Parameter Symbol	Data Path Description		Am29C660C	
			From Input	To Output	Min.	Max.
1	A	tpd	DATA ₀₋₃₁ (Note 3)	SC ₀₋₇		22
	B			DATA ₀₋₃₁ (Note 2)		29
	C			ERROR		21
	D			MULT ERROR		24
2	A	tpd	CB ₀₋₇ (CODE ID 00, 11)	SC ₀₋₇		17
	B			DATA ₀₋₃₁		23
	C			ERROR		16
	D			MULT ERROR		18
3	A	tpd	CB ₀₋₇ (CODE ID 10)	SC ₀₋₇		17
	B			DATA ₀₋₃₁		18
4	A	tpd	GENERATE	SC ₀₋₇		20
	B			ERROR		10
	C			MULT ERROR		12
5		tpd	CORRECT (Not Internal Control Mode)	DATA ₀₋₃₁		17
6	A	tpd	DIAG MODE _{0,1} (Not Internal Control Mode)	SC ₀₋₇		18
	B			DATA ₀₋₃₁		29
	C			ERROR		12
	D			MULT ERROR		23
7	A	tpd	CODE ID _{0,1}	SC ₀₋₇		21
	B			DATA ₀₋₃₁		26
	C			ERROR		20
	D			MULT ERROR		24
8	A	tpd	LE _{IN} (From Latched to Transparent)	SC ₀₋₇		24
	B			DATA ₀₋₃₁		32
	C			ERROR		21
	D			MULT ERROR		25
9		tpd	LE _{OUT} (From Latched to Transparent)	DATA ₀₋₃₁		13
10	A	tpd	LE _{DIAG} (From Latched to Transparent; Not Internal Control Mode)	SC ₀₋₇		18
	B			DATA ₀₋₃₁		27
	C			ERROR		17
	D			MULT ERROR		21

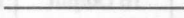




SWITCHING CHARACTERISTICS over MILITARY operating range (Continued)

No		Parameter Symbol	Data Path Description		Am29C660C	
			From Input	To Output	Min.	Max.
11	A	tPD	Internal Control Mode: LE DIAG (From Latched to Transparent)	SC0-7		19
	B			DATA0-31		25
	C			ERROR		18
	D			MULT ERROR		21
12	A	tPD	Internal Control Mode: DATA0-31 (Via Diagnostic Latch)	SC0-7		18
	B			DATA0-31 (Note 2)		29
	C			ERROR		14
	D			MULT ERROR		18
13		tSET	DATA0-31 (Note 4)	LEIN	3	
14		tHOLD			4	
15		tSET	CB0-7 (Note 4)		2	
16		tHOLD			4	
17		tSET	DATA0-31 (Note 4)		7	
18		tHOLD			0	
19		tSET	CB0-7 (Note 4) (CODE ID 00, 11)		10	
20		tHOLD			0	
21		tSET	CB0-7 (Note 4) (CODE ID 10)		10	
22		tHOLD			0	
23		tSET	CORRECT (Note 4)	LEOUT	9	
24		tHOLD			0	
25		tSET	DIAG MODE0, 1 (Note 4)		19	
26		tHOLD			0	
27		tSET	CODE ID0, 1 (Note 4)		12	
28		tHOLD			0	
29		tSET	LEIN (Note 4)		21	
30		tHOLD			0	
31		tSET	DATA0-31 (Note 4)	LEDIAG	3	
32		tHOLD			3	
33		tEN	OE BYTE0-3 (Note 5)	DATA0-31		17
34		tDIS				15
35		tEN	OE sc (Note 5)	SC0-7		18
36		tDIS				15
37		tpw	Minimum Pulse Width: LEIN, LEOUT, LEDIAG		6	

Notes:

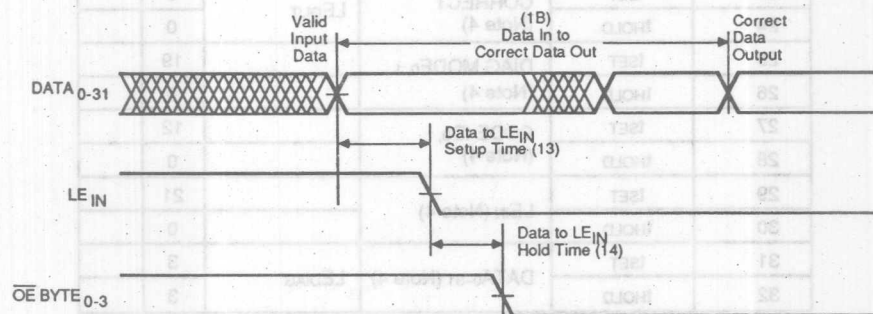
1. $C_L = 50$ pF.
2. Certain parameters are combinational propagation delay calculations.
3. Data In or LEIN to Correct Data Out measurement requires timing as shown in the Switching Waveforms.
4. Setup and Hold times relative to Latch Enables (Latching up data).
5. Output disable tests specified with $C_L = 5$ pF and measured to 0.5 V change of output voltage level. Testing is performed at $C_L = 50$ pF and correlated to $C_L = 5$ pF.

KEY TO SWITCHING WAVEFORMS

WAVEFORM	INPUTS	OUTPUTS
	Must be Steady	Will be Steady
	May Change from H to L	Will be Changing from H to L
	May Change from L to H	Will be Changing from L to H
	Don't Care, Any Change Permitted	Changing, State Unknown
	Does Not Apply	Center Line is High-Impedance "Off" State

KS000010

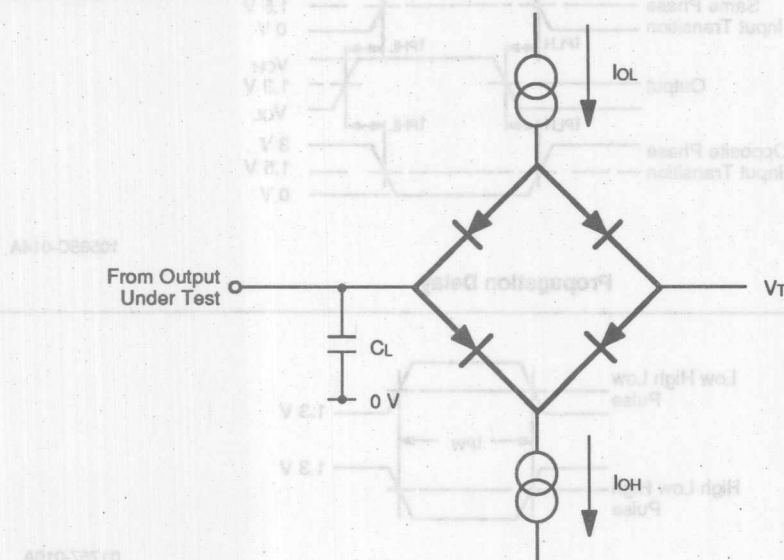
SWITCHING WAVEFORM



10565-017A

DATA₀₋₃₁/LE_{IN} to Correct Data Out

SWITCHING TEST CIRCUIT

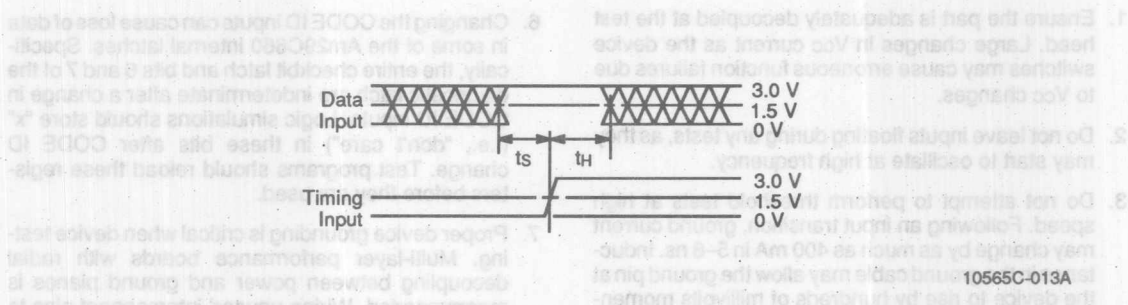


10565C-012A

Notes:

1. $C_L = 50$ pF for all tests except output enable/disable (includes scope probe, wiring and stray capacitance without device in test fixture).
2. $C_L = 5$ pF for output enable/disable tests
3. $V_T = 1.5$ V.

SWITCHING TEST WAVEFORMS

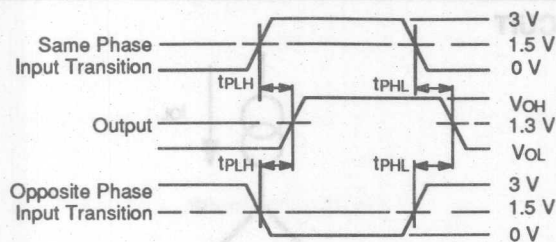


10565C-013A

Notes:

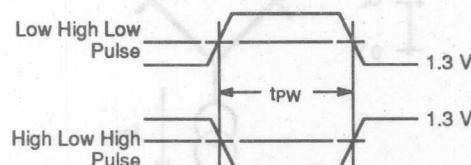
1. Diagram shown for HIGH data only. Output transition may be opposite sense.
2. Cross-hatched area is don't care condition.

Setup and Hold Times



10565C-014A

Propagation Delay



01767-010A

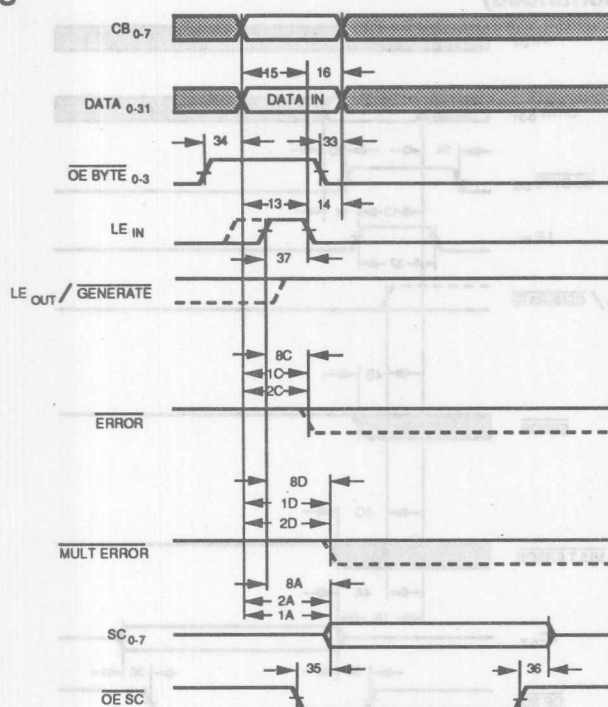
Pulse Width

Notes on Testing

Incoming test procedures on this device should be carefully planned, taking into account the complexity and power levels of the part. The following notes may be useful.

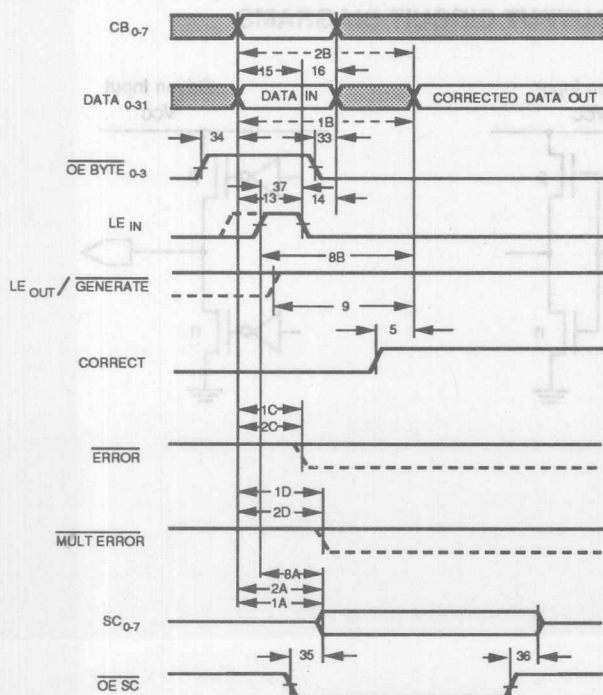
1. Ensure the part is adequately decoupled at the test head. Large changes in V_{CC} current as the device switches may cause erroneous function failures due to V_{CC} changes.
2. Do not leave inputs floating during any tests, as they may start to oscillate at high frequency.
3. Do not attempt to perform threshold tests at high speed. Following an input transition, ground current may change by as much as 400 mA in 5–8 ns. Inductance in the ground cable may allow the ground pin at the device to rise by hundreds of millivolts momentarily.
4. Use extreme care in defining input levels for AC tests. Many inputs may be changed at once, so there will be significant noise at the device pins and they may not actually reach V_{IL} or V_{IH} until the noise has settled. AMD recommends using $V_{IL} \leq 0$ V and $V_{IH} \geq 3$ V for AC tests.
5. To simplify failure analysis, programs should be designed to perform DC, Function, and AC tests as three distinct groups of tests.
6. Changing the CODE ID inputs can cause loss of data in some of the Am29C660 internal latches. Specifically, the entire checkbit latch and bits 6 and 7 of the diagnostic latch are indeterminate after a change in CODE ID inputs. Logic simulations should store "x" (i.e., "don't care") in these bits after CODE ID change. Test programs should reload these registers before they are used.
7. Proper device grounding is critical when device testing. Multi-layer performance boards with radial decoupling between power and ground planes is recommended. Wiring unused interconnect pins to the ground plane is recommended. The ground plane must be sustained from the performance board to the device under test interface board. To minimize inductance, heavy-gauge stranded wire with twisted pairs should be used for power wiring.

TIMING DIAGRAMS



10565C-015A

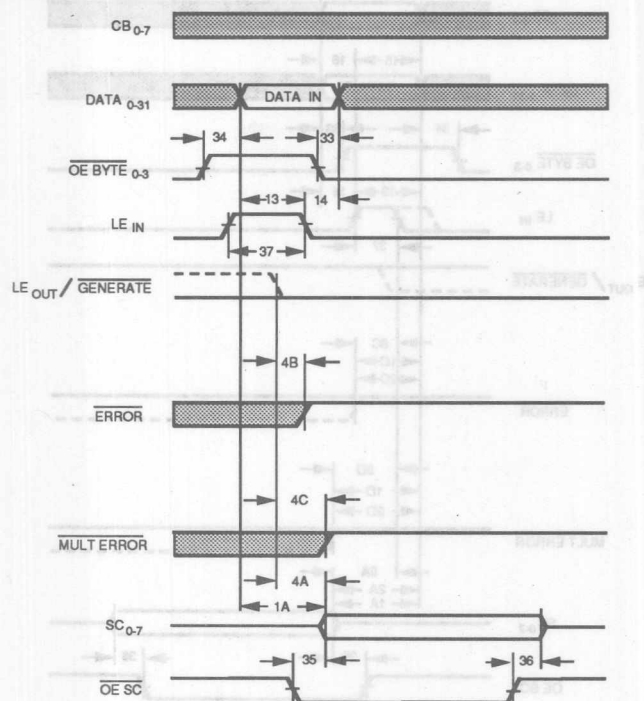
Detect Mode



10565C-016A

Correct Mode

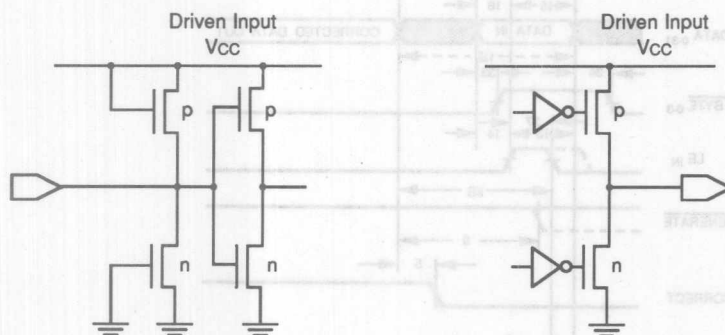
TIMING DIAGRAMS (Continued)



Generated Mode

10565C-017A

EQUIVALENT INPUT/OUTPUT CIRCUIT DIAGRAMS



10565C-018A

Am29C668

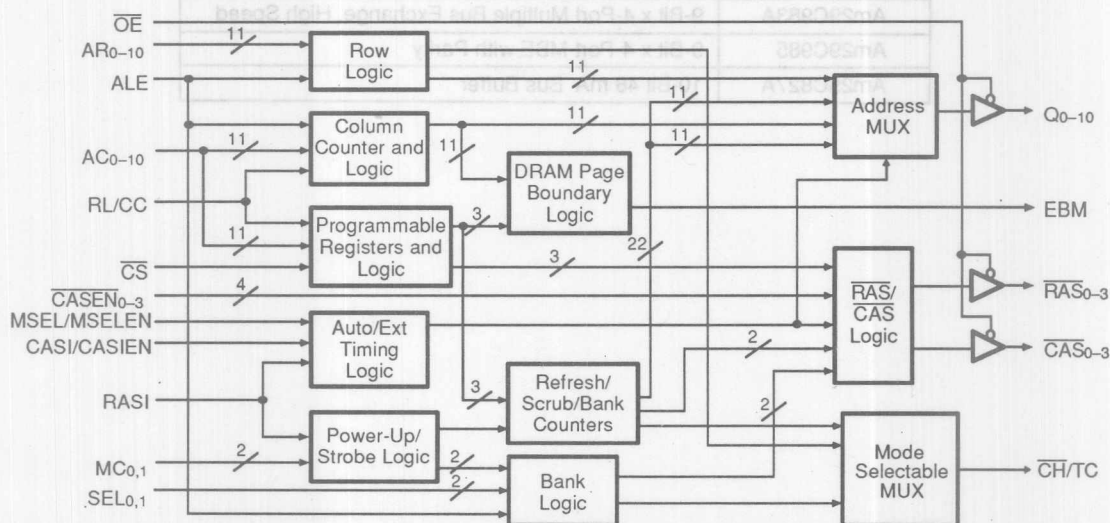
4M Configuration Dynamic Memory Controller/Driver

**Advanced
Micro
Devices**

DISTINCTIVE CHARACTERISTICS

- Provides control for 4M, 1M, 256K, 64K dynamic RAMs
- Programmable Burst/Block Access support for Am29000, 68000 family, iAPX family
- Proprietary "Cache" Mode supports Page Mode accessing
- Nibble mode support (for Page Mode or Nibble Mode DRAMs)
- Selectable Address and Strobe autotiming or external timing
- Supports "Scrubbing" with refresh when used in an EDC system
- Supports CAS before RAS refresh
- Byte and Bank CAS Decoding
- Selectable 2 or 4 bank drive
- Outputs directly drive up to 88 DRAMs, with a guaranteed worst-case limit on the undershoot and overshoot
- Low-power advanced sub-micron CMOS process
- User configurable to replace Am2968A and Am29368 DMCs

BLOCK DIAGRAM



11068-001C

GENERAL DESCRIPTION

The Am29C668 4M Configurable Dynamic Memory Controller/Driver (CDMC) is designed for high performance memory systems. The CDMC acts as the address controller between the processor and the dynamic memory array. It uses its 11-bit row latch and 11-bit-column latch and counter to hold the row and column addresses, respectively, for multiplexing these to any DRAM size up to 4M. These latches and counter and the row/column refresh counter are used to directly drive the address lines of the DRAM array. The output of the 2-bit bank latch is decoded to select the bank to be accessed.

The Am29C668 has two basic modes of operation, read/write and refresh. In the read/write mode, the Am29C668 latches the row, column, and bank addresses and multiplexes them to the DRAM array. This

multiplexing occurs under the control of the internally-generated timing strobes in the Auto Timing Mode, or the externally-generated MSEL in External Timing Mode. The read/write mode of the Am29C668 may be optimized for the shortest memory access time, through burst/block access, "cache" mode access or nibble mode access.

In the refresh mode, the refresh address is generated by the Am29C668 refresh counter. This counter is automatically adjusted for different DRAM sizes. If memory scrubbing is not being implemented, only the row counter is used to generate the row address for refresh. When memory scrubbing is being performed in EDC systems, both the row and column address counters are used.

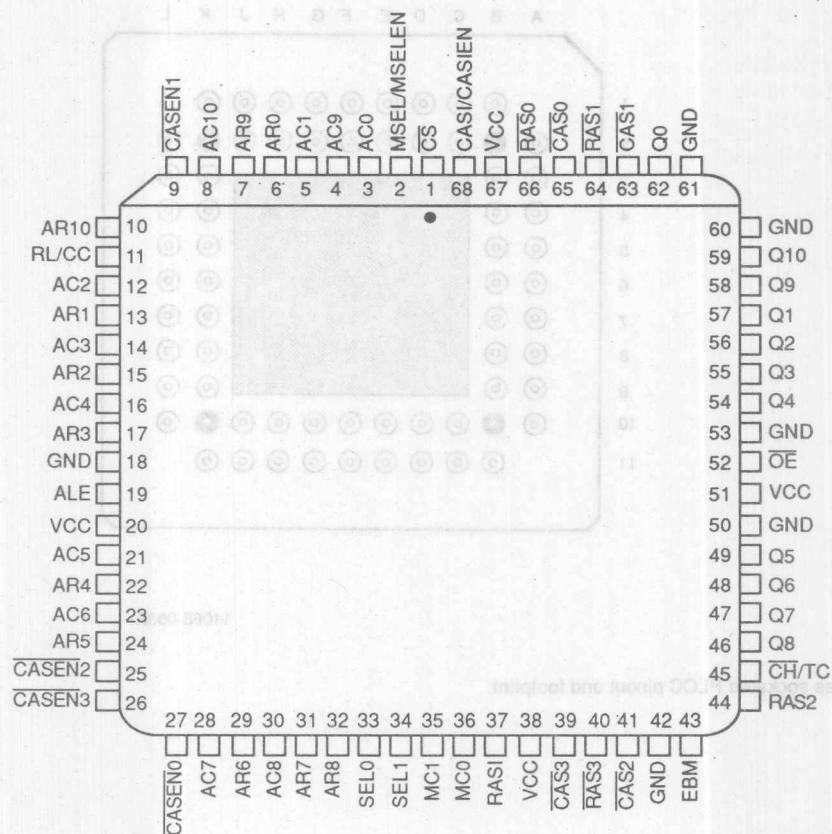
RELATED AMD PRODUCTS

Part No.	Description
Am29C60A	High Speed CMOS Cascadable 16-Bit EDC
Am29C660E	9 ns CMOS Cascadable 32-Bit EDC
Am29C676	11-Bit Driver for 4M x 1 and 4M x 4 DRAMs
Am29C983	9-Bit x 4-Port Multiple Bus Exchange
Am2965/6	8-Bit Dynamic RAM Driver Inverting/Non-Inverting
Am29C983A	9-Bit x 4-Port Multiple Bus Exchange, High Speed
Am29C985	9-Bit x 4-Port MBE with Parity
Am29C827A	10-Bit 48 mA Bus Buffer

CONNECTION DIAGRAM

Top View

PLCC



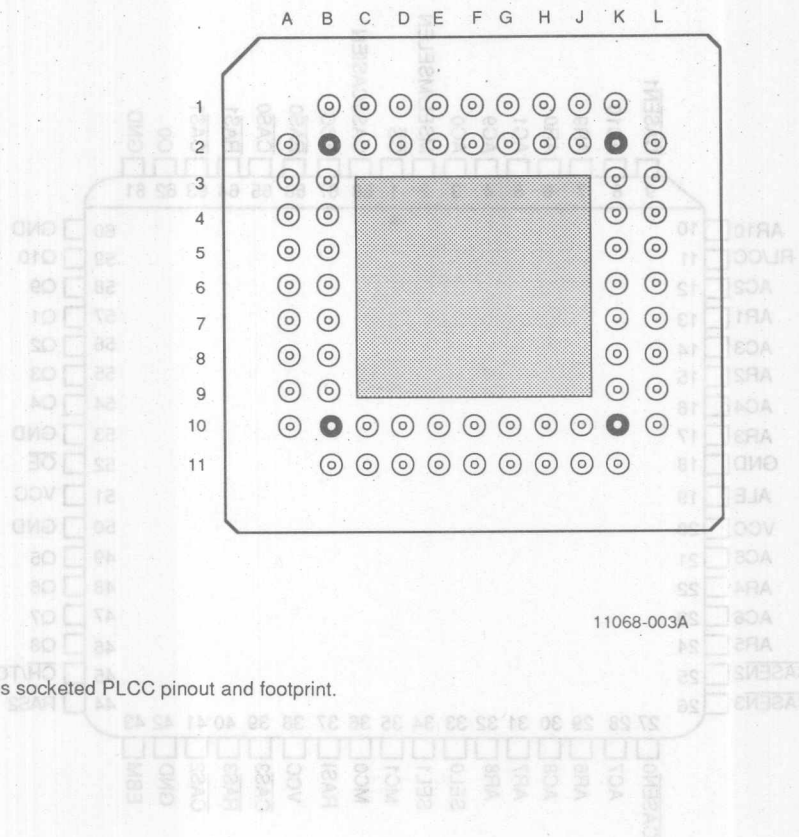
11068-002A

Note:

Pin 1 is marked for orientation (PLCC only).

CONNECTION DIAGRAM **Top View (Pins Pointing Down)**

PGA*



*Pinout matches socketed PLCC pinout and footprint.



PGA PIN DESIGNATIONS

(Sorted by Pin Number)

Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.
AR ₁₀	A-2	CASEN ₂	B-9	MC ₀	F-10	Q ₁	K-4
AC ₂	A-3	AC ₇	B-10	MC ₁	F-11	Q ₃	K-5
AC ₃	A-4	CASEN ₀	B-11	V _{CC}	G-1	GND	K-6
AC ₄	A-5	AR ₉	C-1	CASI/CASIEN	G-2	V _{CC}	K-7
GND	A-6	AC ₁₀	C-2	V _{CC}	G-10	Q ₅	K-8
V _{CC}	A-7	AC ₈	C-10	RASI	G-11	Q ₇	K-9
AR ₄	A-8	AR ₆	C-11	CAS ₀	H-1	CH/TC	K-10
AR ₅	A-9	AC ₁	D-1	RAS ₀	H-2	EBM	K-11
CASEN ₃	A-10	AR ₀	D-2	RAS ₃	H-10	GND	L-2
CASEN ₁	B-1	AR ₈	D-10	CAS ₃	H-11	Q ₉	L-3
RL/CC	B-2	AR ₇	D-11	CAS ₁	J-1	Q ₂	L-4
AR ₁	B-3	AC ₀	E-1	RAS ₁	J-2	Q ₄	L-5
AR ₂	B-4	AC ₉	E-2	GND	J-10	OE	L-6
AR ₃	B-5	SEL ₁	E-10	CAS ₂	J-11	GND	L-7
ALE	B-6	SEL ₀	E-11	GND	K-1	Q ₆	L-8
AC ₅	B-7	CS	F-1	Q ₀	K-2	Q ₈	L-9
AC ₆	B-8	MSEL/MSELEN	F-2	Q ₁₀	K-3	RAS ₂	L-10

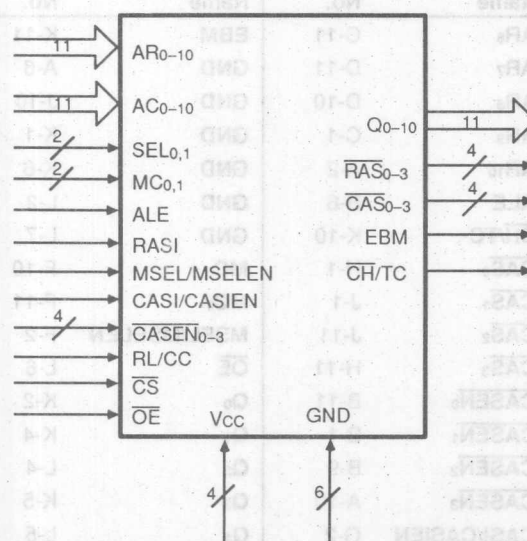
(Sorted by Pin Name)

Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.	Pin Name	Pin No.
AC ₀	E-1	AR ₆	C-11	EBM	K-11	Q ₆	L-8
AC ₁	D-1	AR ₇	D-11	GND	A-6	Q ₇	K-9
AC ₂	A-3	AR ₈	D-10	GND	J-10	Q ₈	L-9
AC ₃	A-4	AR ₉	C-1	GND	K-1	Q ₉	L-3
AC ₄	A-5	AR ₁₀	A-2	GND	K-6	Q ₁₀	K-3
AC ₅	B-7	ALE	B-6	GND	L-2	RAS ₀	H-2
AC ₆	B-8	CH/TC	K-10	GND	L-7	RAS ₁	J-2
AC ₇	B-10	CAS ₀	H-1	MC ₀	F-10	RAS ₂	L-10
AC ₈	C-10	CAS ₁	J-1	MC ₁	F-11	RAS ₃	H-10
AC ₉	E-2	CAS ₂	J-11	MSEL/MSELEN	F-2	RASI	G-11
AC ₁₀	C-2	CAS ₃	H-11	OE	L-6	RL/CC	B-2
AR ₀	D-2	CASEN ₀	B-11	Q ₀	K-2	SEL ₀	E-11
AR ₁	B-3	CASEN ₁	B-1	Q ₁	K-4	SEL ₁	E-10
AR ₂	B-4	CASEN ₂	B-9	Q ₂	L-4	V _{CC}	A-7
AR ₃	B-5	CASEN ₃	A-10	Q ₃	K-5	V _{CC}	G-1
AR ₄	A-8	CASI/CASIEN	G-2	Q ₄	L-5	V _{CC}	G-10
AR ₅	A-9	CS	F-1	Q ₅	K-8	V _{CC}	K-7

PGA to PLCC PIN CONVERSION

A2	10	B9	25	F10	36	K4	57
A3	12	B10	28	F11	35	K5	55
A4	14	B11	27	G1	67	K6	53
A5	16	C1	7	G2	68	K7	51
A6	18	C2	8	G10	38	K8	49
A7	20	C10	30	G11	37	K9	47
A8	22	C11	29	H1	65	K10	45
A9	24	D1	5	H2	66	K11	43
A10	26	D2	6	H10	40	L2	60
B1	9	D10	32	H11	39	L3	58
B2	11	D11	31	J1	63	L4	56
B3	13	E1	3	J2	64	L5	54
B4	15	E2	4	J10	42	L6	52
B5	17	E10	34	J11	41	L7	50
B6	19	E11	33	K1	61	L8	48
B7	21	F1	1	K2	62	L9	46
B8	23	F2	2	K3	59	L10	44

LOGIC SYMBOL



Die Size: 0.233" x 0.165"

Gate Count: 3600

11068-004B

Package Information

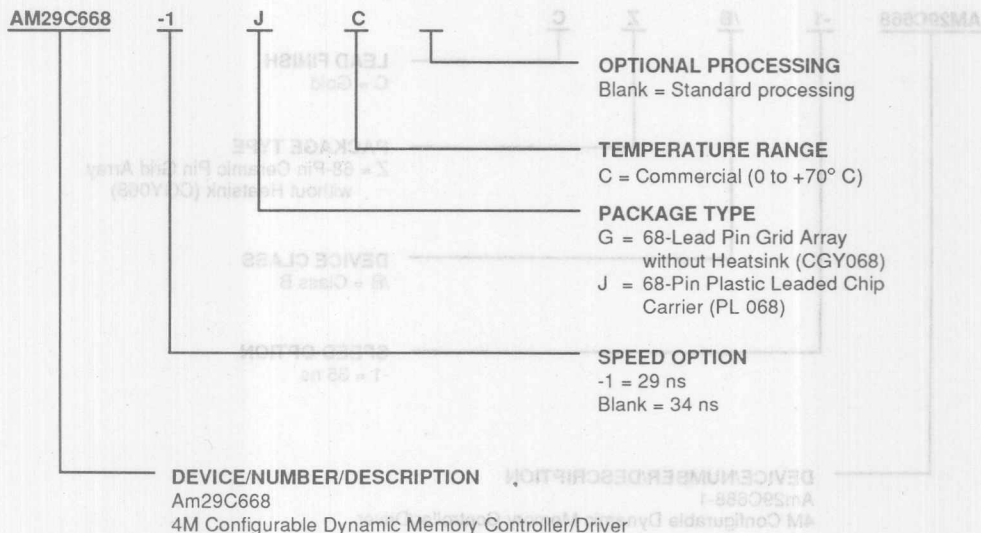
Parameter	PGA	PLCC	Units
θ_{JA}	34	35	°C/W

ORDERING INFORMATION

Standard Products

AMD standard products are available in several packages and operating ranges. The ordering number (Valid Combination) is formed by a combination of these elements:

Device Number
Speed Option (if applicable)
Package Type
Temperature Range
Optional Processing



Valid Combinations	
AM29C668	JC, GC
AM29C668-1	

Valid Combinations

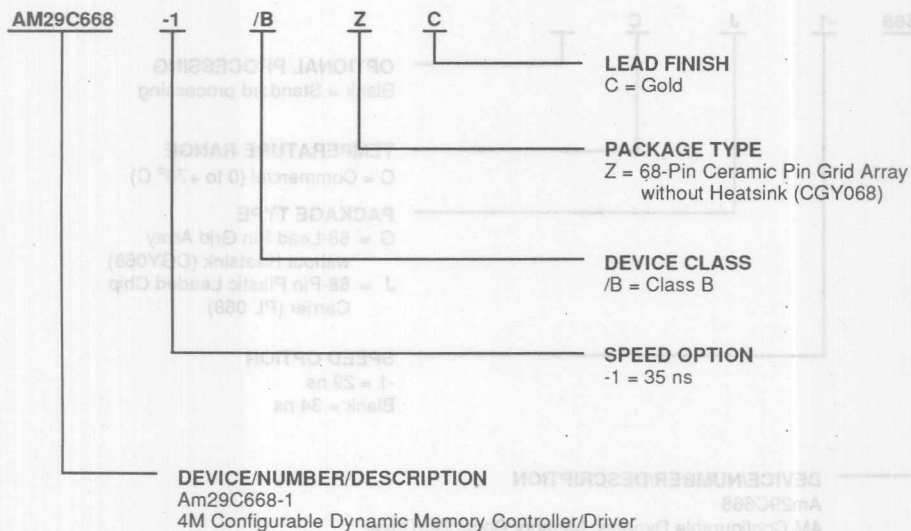
Valid Combinations list configurations planned to be supported in volume for this device. Consult the local AMD sales office to confirm availability of specific valid combinations, to check on newly released combinations, and to obtain additional data on AMD's standard military grade products.

ORDERING INFORMATION

APL Products

AMD standard products are available in several packages and operating ranges. The ordering number (Valid Combination) is formed by a combination of these elements:

Device Number
Speed Option (if applicable)
Device Class
Package Type
Lead Finish



Valid Combinations

AM29C668-1	/BZC
------------	------

Valid Combinations

Valid Combinations list configurations planned to be supported in volume for this device. Consult the local AMD sales office to confirm availability of specific valid combinations, to check on newly released combinations, and to obtain additional data on AMD's standard military grade products.

Group A Tests

Group A tests consist of Subgroups
1, 2, 3, 7, 8, 9, 10, 11

PIN DESCRIPTION

AC₀₋₁₀ and AR₀₋₁₀

Column and Row Address Inputs (Inputs(22))

The address on these lines is latched by the LOW going edge of the Address Latch Enable (ALE) signal. AC₀₋₁₀ are connected to the lower side of the system address bus and are driven on the output address lines Q₀₋₁₀ when the MSEL (Multiplexer Select) signal is HIGH. AR₀₋₁₀ are connected to the upper side of the system address bus and are driven on the output address lines Q₀₋₁₀ when the MSEL signal is LOW.

AC₀₋₇ and AR₀₋₇ are used for 64K DRAMs.

AC₀₋₈ and AR₀₋₈ are used for 256K DRAMs.

AC₀₋₉ and AR₀₋₉ are used for 1M DRAMs.

AC₀₋₁₀ and AR₀₋₁₀ are used for 4M DRAMs.

ALE

Address Latch Enable (Input; Active HIGH)

This input causes the Row Latch, Column Latch and Counter, and the Bank Latch to become transparent allowing the latches to accept new input data. A LOW input on ALE latches the input data, assuming it meets specified set-up and hold requirements.

CH/TC

Cache Hit/Terminal Count

(Outputs; Active LOW/HIGH)

This is a dual function output, dependent upon the Mode Control inputs in the Configuration Register. If the Mode Control inputs are 1,0 (MC_{0,1} = 10) the Am29C668 is in the Refresh-With-Scrubbing or Initialize mode and this output acts as the Terminal Count (TC).

Cache Hit (CH) (if bit #6 in the configuration register is set to "1")

This output goes active (LOW) when the current memory access is to the same row and the same bank as the previous access. The CH signal is used to facilitate Fast Page Mode or Static Column accesses.

As **Terminal Count**, this output goes active (HIGH) when the Refresh Counter has gone through an entire count. The Refresh Counter is user configured for DRAM size (64K, 256K, 1M, 4M) and number of banks (2 or 4), which are programmable via the Memory Size Bits and the RAS/CAS Configuration Bit, respectively, in the Configuration Register (Reference Figure 6). The TC signal is used to indicate the end of initialization in an Error Detection and Correction (EDC) system.

CAS₀₋₃

Column Address Strobe (Outputs (4); Active LOW; Three State)

Each $\overline{\text{CAS}}_n$ output will go active when selected by SEL_{0,1} in a bank-wise $\overline{\text{CAS}}$ decoding method (CDM = 0, reference Figure 6) or when selected by CASEN₀₋₃ in a byte-wise $\overline{\text{CAS}}$ decoding implementation. This will occur only when CASI goes active in the External Timing

Mode or when CASIEN and the internally generated $\overline{\text{CAS}}$ go active in the Auto Timing Mode.

Each output provides a $\overline{\text{CAS}}_n$ signal to one of four banks of the dynamic memory, if four banks are used. If two banks are used, each bank can use 2 $\overline{\text{CAS}}_n$ signals to reduce the capacitive load on each. The number of banks (2 or 4) is programmable via the RAS/ $\overline{\text{CAS}}$ Configuration Bit in the Configuration Register (reference Table 4).

The $\overline{\text{CAS}}_n$ outputs contain pull-up resistors which ensure a logical HIGH (inactive) when in the high impedance state.

CASEN₀₋₃

Column Address Strobe Enable

(Inputs (4); Active LOW)

When a byte-wise method is used for $\overline{\text{CAS}}$ decoding these four inputs are decoded externally to handle byte operations. The timing generation may be Auto or External. Only those $\overline{\text{CAS}}_n$ outputs will be activated whose corresponding CASEN_n inputs are selected by the external byte decode circuit.

When a bank-wise method is used for $\overline{\text{CAS}}$ decoding these inputs are not used.

CASI/CASIEN

Column Address Strobe Input/Column Address Strobe Input Enable (Input; Active HIGH)

This is a dual function input. In the External Timing mode this input is used as CASI. With a bank-wise $\overline{\text{CAS}}$ decoding method, the internally decoded $\overline{\text{CAS}}_n$ output is forced LOW after CASI goes active. When used as CASI with a byte-wise decoding method, the selected $\overline{\text{CAS}}_n$ output is forced LOW depending upon the externally decoded CASEN_n inputs after CASI goes active.

In Auto Timing Mode this input is used as CASIEN. With a bank-wise $\overline{\text{CAS}}$ decoding method, the decoded $\overline{\text{CAS}}_n$ output is forced LOW, if both the internally generated $\overline{\text{CAS}}$ and the CASIEN signals are active. This input is used to delay the $\overline{\text{CAS}}_{0-3}$ outputs from going active if desired, resulting in a longer auto timing access sequence. This input is generally not used as CASIEN with a byte-wise $\overline{\text{CAS}}$ decoding implementation.

$\overline{\text{CS}}$

Chip Select (Input; Active LOW)

This input is used to enable the Am29C668. When active, the Am29C668 operates normally in all four modes. When $\overline{\text{CS}}$ goes inactive (HIGH), the device will not enter the Read/Write mode.

EBM

End Burst/Block Mode (Output Active HIGH)

This output is only used in the burst/block mode of data transfer. It indicates to the processor that the Am29C668 cannot perform any more data transfers in

the burst/block mode for one of two reasons. Either the DRAM page boundary is reached (in which case a new row address is required from the processor), or a programmed allowable number of transfers has been completed.

GND (6) 0 V Power Supply

These pins are the 0 V power supply for the Am29C668. All grounds must be connected for proper device operation.

MC_{0,1} Mode Control (Inputs (2))

These inputs specify one of four modes of operation of the Am29C668. Operating modes are described in Table 1.

MSEL/MSELEN

Multiplexer Select/Multiplexer Select Enable (Input; Active HIGH)

This is a dual function input. In the External Timing mode this input is used as **MSEL**. When MSEL is HIGH the column address is selected. When MSEL is LOW the row address is selected.

In the Auto Timing Mode this input acts as **MSELEN**. When MSELEN is HIGH and the internally generated MSEL is active the column address is selected. When MSELEN is LOW or the internally generated MSEL is inactive the row address is selected. MSELEN is used to delay the address change from row to column, if desired, resulting in a longer auto timing access sequence.

The address may come from either the address latches and counter or the refresh address counter depending upon MC_{0,1}. The MSEL/MSELEN input is only applicable in the Read/Write or Refresh with Scrubbing Modes.

OE

Output Enable (Input; Active LOW)

This input enables/disables the output signals. When OE is inactive (HIGH), all address outputs of the Am29C668 enter a high impedance state and the \overline{RAS}_n and \overline{CAS}_n outputs are pulled inactive (HIGH).

Q₀₋₁₀

Address Outputs (Outputs(11); Three State)

These edge rate controlled outputs drive the dynamic memory address inputs. The drivers on these lines are able to drive high capacitive loads, which are specified at 350pF. Greater capacitive loads may also be driven, however. See section labeled "Typical Change in Propagation Delay vs Loading Capacitance" following the AC Characteristics.

RAS_{0,3}

Row Address Strobe (Outputs (4); Active LOW; Three State)

Each Row Address Strobe output provides a \overline{RAS}_n signal to one of four memory banks. Each will go low when selected by SEL_{0,1} and only when RASI goes HIGH. All four go LOW in response to RASI in the refresh modes.

When a 2 bank $\overline{RAS}/\overline{CAS}$ configuration is selected (RCC = 1), \overline{RAS}_0 and \overline{RAS}_1 are tied together internally, as are \overline{RAS}_2 and \overline{RAS}_3 . This reduces the capacitive loading on the \overline{RAS}_n outputs in a two bank system (reference Table 4).

In four bank mode, the \overline{RAS}_n outputs are decoded with SEL_{0,1}. In two bank mode these outputs are decoded with SEL₀. In this case SEL₁, should be tied LOW.

The \overline{RAS}_n outputs contain pull-up resistors which ensure a logical HIGH (inactive) when in the high impedance state.

RASI

Row Address Strobe Input (Input; Active HIGH)

During normal memory cycles, the decoded \overline{RAS}_n outputs ($\overline{RAS}_0, \overline{RAS}_1, \overline{RAS}_2, \overline{RAS}_3$) as determined by SEL_{0,1} and the RCC bit in the Configuration Register are forced LOW after RASI goes active HIGH. During refresh, all four \overline{RAS}_n outputs go LOW after RASI goes active HIGH. If auto timing is enabled, the HIGH going edge of RASI also initiates the internal timing cycle and its LOW going edge terminates the internal timing cycle.

RL/CC

Register Load/Column Clock (Input)

This is a dual function pin which depends upon the Mode Control inputs (MC_{0,1}). If MC_{0,1} = 11, the Am29C668 is in the Reset Mode and this pin acts as the Register Load signal. If MC_{0,1} = 01 the Am29C668 is in the Read/Write Mode and this input acts as the Column Clock signal.

When used as **Register Load**, the LOW-to-HIGH edge of the signal loads either the Burst Count Register, the Mask Register, or the Configuration Register via the AC₀₋₁₀ Address Inputs. (Reference Figure 5).

When used as **Column Clock**, the HIGH-to-LOW edge of the signal increments the Column Counter during burst and nibble mode accessing.

SEL_{0,1}

Bank Select (Inputs (2))

These two inputs are the highest-order address bits when the Am29C668 is used in the normal access mode or in the burst/block access mode. SEL_{0,1} is used in the Read/Write Mode to select which bank of memory will receive the \overline{RAS}_n and \overline{CAS}_n signals when RASI and CASI (or the internally generated CAS in the auto-timing mode) go active HIGH. The \overline{CAS}_n signals will not be decoded from SEL_{0,1} if a byte-wise \overline{CAS} decoding scheme is selected. In two bank mode, only SEL₀ is used. SEL₁ should be tied LOW.

V_{CC} (4) + 5 V

Positive Power Supply Voltage

These inputs provide the power necessary to operate the Am29C668. All power supply inputs must be connected for proper device operation.

Table 1. Mode Control Function *

MC ₀	MC ₁	Operating Mode
0	0	<p>Refresh Without Scrubbing (a more detailed description can be found in the section entitled "Refresh Modes")</p> <p>a) $\overline{\text{RAS}}$-Only Refresh: Refresh cycles are performed with only the row refresh counter being used to increment addresses. In this mode, all four $\overline{\text{RAS}}_n$ outputs are active while the four $\overline{\text{CAS}}_n$ outputs are held inactive.</p> <p>b) $\overline{\text{CAS}}$ Before $\overline{\text{RAS}}$ Refresh: Refresh addresses are generated internally by the DRAMs. In this mode, all four $\overline{\text{CAS}}_n$ outputs are active followed by all four $\overline{\text{RAS}}_n$ outputs going active. This new type of refresh is selected via the CBR-bit in the Configuration Register. In this mode, RAS_I controls the $\overline{\text{CAS}}_n$ outputs and CAS_I controls the $\overline{\text{RAS}}_n$ outputs.</p>
1	0	<p>Refresh With Scrubbing/Initialize (EDC Systems)</p> <p>This mode may be used only in systems with Error Detection and Correction (EDC) capability. In this mode, refresh cycles are performed with both the row and column refresh counters generating the addresses. MSEL is used to select between the row and column addresses. All four $\overline{\text{RAS}}_n$ signals go active in response to RAS_I and one $\overline{\text{CAS}}_n$ output goes active in response to CAS_I. The CAS_n output is decoded from the bank refresh counter. The remaining three $\overline{\text{CAS}}_n$ outputs are left inactive, while their respective banks undergo normal refresh. This mode is also used to initialize the memory array by writing a known data pattern and corresponding check bits.</p>
0	1	<p>Read/Write</p> <p>This mode is used to perform read/write operations. The row address is taken from the row latch and the column address is taken from the column latch and counter. SEL_{0,1} are decoded to determine which $\overline{\text{RAS}}_n$ and CAS_n will be active.</p>
1	1	<p>Reset/Configuration</p> <p>This mode is used to clear the refresh counters and the Register Logic. These operations are performed on the HIGH-to-LOW transition of RAS_I. This mode is used to load the configuration, burst count, and mask registers.</p>

Table 2. Address Output Function

\overline{CS}	MC ₁	MC ₀	Internal MSEL	Mode	Address Multiplexer Output
0	0	0	X	Refresh W/O Scrubbing	Row Counter
	0	1	1	Refresh with Scrubbing	Column Counter
			0		Row Counter
	1	0	1	Read/Write	Column Latch
			0		Row Latch
	1	1	X	Reset/Configuration	All Zero
1	0	0	X	Refresh W/O Scrubbing	Row Counter
	0	1	1	Refresh with Scrubbing	Column Counter
			0		Row Counter
	1	0	X	Read/Write	All Zero
	1	1	X	Reset	All Zero

X = Don't care

This mode is used to clear the refresh counter and the Register Latch. These operations are performed on the HIGH-to-LOW transition of RAS. This mode is used to load the configuration, burst count, and mask registers.					
Reset/Configuration					
This mode is used to determine which RAS and CAS will be active.					
This mode is used to perform read/write operations. The row address is taken from the row latch and the column address is taken from the column latch and counter. RAS and CAS will be active.					
This mode may be used in systems with Error Detection and Correction (EDC).					

Table 3. $\overline{\text{RAS}}$ Output Function

Inputs		Outputs							
Internal RAS _I	$\overline{\text{CS}}$	MC	SEL*	RCC**	MODE	$\overline{\text{RAS}}_n$			
		1 0	1 0			3	2	1	0
0	X	X X	X X	X	No operation	1	1	1	1
1	0	0 0	X X	X	Refresh W/O Scrubbing	0	0	0	0
		0 1	X X	X	Refresh with Scrubbing	0	0	0	0
		1 0	0 0	0	Read/Write	1	1	1	0
			X 0	1		1	1	0	0
			0 1	0		1	1	0	1
			X 1	1		0	0	1	1
			1 0	0		1	0	1	1
			X 0	1		1	1	0	0
			1 1	0		0	1	1	1
			X 1	1		0	0	1	1
		1 1	X X	X	Reset/Configuration	0	0	0	0
		1	0 0	X X	X	Refresh W/O Scrubbing	0	0	0
	0 1		Refresh with Scrubbing			0	0	0	0
	1 0		Read/Write			1	1	1	1
	1 1		Reset			0	0	0	0

* After Internal RAS_I is asserted, changing SEL_{0,1} will not effect the $\overline{\text{RAS}}_n$ decoding until Internal RAS_I is deasserted.

** Reference Figure 6.

Table 4. $\overline{\text{RAS}}/\overline{\text{CAS}}$ Configuration Decode*

RCC	Mode	$\overline{\text{RAS}}/\overline{\text{CAS}}$ CONFIGURATION		
0	4-Bank	RAS ₀	CAS ₀	BANK 0
		RAS ₁	CAS ₁	BANK 1
		RAS ₂	CAS ₂	BANK 2
		RAS ₃	CAS ₃	BANK 3
1	2-Bank	RAS ₀	CAS ₀	BANK 0
		RAS ₁	CAS ₁	BANK 1
		RAS ₂	CAS ₂	BANK 1
		RAS ₃	CAS ₃	

*CDM = 0

Table 5. $\overline{\text{CAS}}$ Output Function

Inputs								Outputs			
CDM**	Internal CASI†	$\overline{\text{CS}}$	CBR**	MC	SEL*	Internal Counter	RCC**	$\overline{\text{CAS}}_n$			
				1 0	1 0	1 0		3	2	1	0
			1					0	0	0	0
			0	0 0	X X	X X	X	1	1	1	1
			1		X X	X X	X	1	1	1	1
						0 0	0	1	1	1	0
						0 1	1	1	1	0	0
			0	0 1	X X	0 1	0	1	1	0	1
						1 0	1	0	0	1	1
						1 1	0	1	0	1	1
							1	1	1	0	0
							0	0	1	1	1
0	1	0					1	0	0	1	1
					0 0		0	1	1	1	0
					0 1		1	1	1	0	0
					1 0	X X	1	0	0	1	1
					1 1		0	1	0	1	1
							1	1	1	0	0
							0	0	1	1	1
							1	0	0	1	1
				1 1	X X	X X	X	1	1	1	1
			1	0 0	X X	X X	X	0	0	0	0
			0		X X	X X	X	1	1	1	1
			1		X X	X X	X	1	1	1	1
						0 0	0	1	1	1	0
						0 1	1	1	1	0	0
						1 0	0	1	0	1	1
						1 1	1	1	1	0	0
							0	0	1	1	1
							1	0	0	1	1
				1 0	X X	X X	X	1	1	1	1
				1 1							
0	X	X	X	X X	X X	X X	X	1	1	1	1

† In the external timing mode, Internal CASI follows the CASI input. In Autotiming Mode, this signal is generated internally and is enabled by the CASIEN inputs.

* After Internal RAS₁ is asserted, changing SEL_{0,1} will not effect the $\overline{\text{CAS}}_n$ decoding until Internal RAS₁ is deasserted.

** Reference Figure 6. For CDM = 1, $\overline{\text{CASEN}} = 3$ will enable their respective $\overline{\text{CAS}}_0 = 3$ in read/write mode.

FUNCTIONAL DESCRIPTION (Note 1)

General Description

The Am29C668 4M Configurable Dynamic Memory Controller/Driver provides the controls required to operate dynamic RAMs up to 4Mbit x n. Manufactured in sub-micron CMOS technology, the Am29C668 performs the address control and generation function and strobe control and generation for 64K, 256K, 1M or 4M DRAMs. The Am29C668 controls the address to the DRAMs from the processor in the read/write mode and it generates and controls the address to the DRAMs in the refresh mode. The Am29C668 also generates the row and column address strobe signals in the read/write and refresh modes.

The Am29C668 has on-chip series damping resistors on its driver outputs to restrict the output signals to +0.8-V overshoot and -1.0V undershoot maximum (See Switching Waveforms).

Logic Overview

The functional blocks of the Am29C668 can be summarized as follows (reference block diagram):

- Row Logic
- Column Counter and Logic
- Bank Logic
- Programmable Registers and Logic
- Auto/External Timing Logic
- Power-Up/Strobe Logic
- DRAM Page Boundary Logic
- Refresh/Scrubbing/Bank Counters
- Address Multiplexer
- RAS/CAS Logic

Row Logic

This block (Figure 1) consists of a Row Latch, a Register, and a Comparator. The 11-bit Row Latch holds the DRAM row address. It is transparent when the Address Latch Enable signal is HIGH, and the address is latched on the LOW-going edge of ALE.

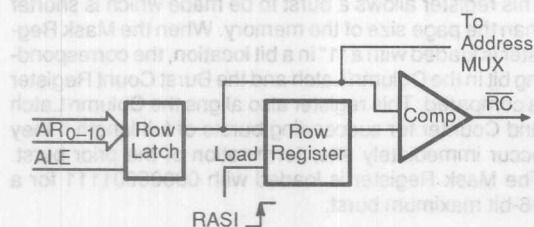


Figure 1. Row Logic

Note:

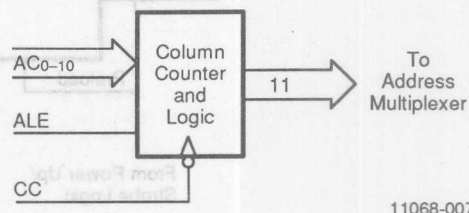
1. Modes of operation are described beginning on page 22.

The 11-bit Row Register holds the row address of the previous DRAM access. The register is clocked at the beginning of every access where $MC_{0,1} = 01$ by the HIGH-going edge of the RASi input.

The Row and Bank Comparator compares the row and bank addresses of the current access with the row address of the previous access and generates the Cache Hit (\overline{CH}) signal. \overline{CH} is LOW if the current row and bank addresses are the same as the previous row and bank addresses, respectively. \overline{CH} is high if they are not. This indicator is used by the external timing generator during Cache Mode accesses. The RASi input is held active (HIGH) if consecutive accesses are to the same row in the same bank, saving precharge time and access time on the current RAS_n . The RASi input is deactivated if consecutive accesses are to different rows or banks, thereby ending the "cache" access.

Column Counter and Logic

The block (Figure 3) consists of the Column Latch and Counter. The 11-bit loadable counter holds the DRAM column address. The counter is transparent when ALE is HIGH and the address is loaded on the LOW-going edge of ALE. The HIGH-to-LOW edge of the signal increments the Column Counter. ALE must be LOW in order to increment the counter.



11068-007A

Figure 3. Column Counter and Logic

If the Nibble Count bit of the Configuration Register is enabled ($NIBCNT = 1$), then only the two LSBs of the Column Latch are clocked, generating a modulo four nibble count (Reference Nibble Mode section).

Programmable Registers and Logic

This block (Figure 5) consists of the Configuration Register, Burst Count Register, Mask Register, Column Comparator Logic, Register Load Logic, and DRAM Size Decoder.

In order to load the 11-bit Configuration Register, a device reset ($MC_0,1=11$ with RAS_1) must occur followed

by switching MC_0 , or MC_1 to 0 (to end reset operation). Then the Configuration Register is loaded via the column address bus (AC_0-10) by the High-going edge of RL/CC signal (F) with $MC_0,1=11$. The Configuration Register is programmed to select the options shown in Figure 6 (reference Figure 15).

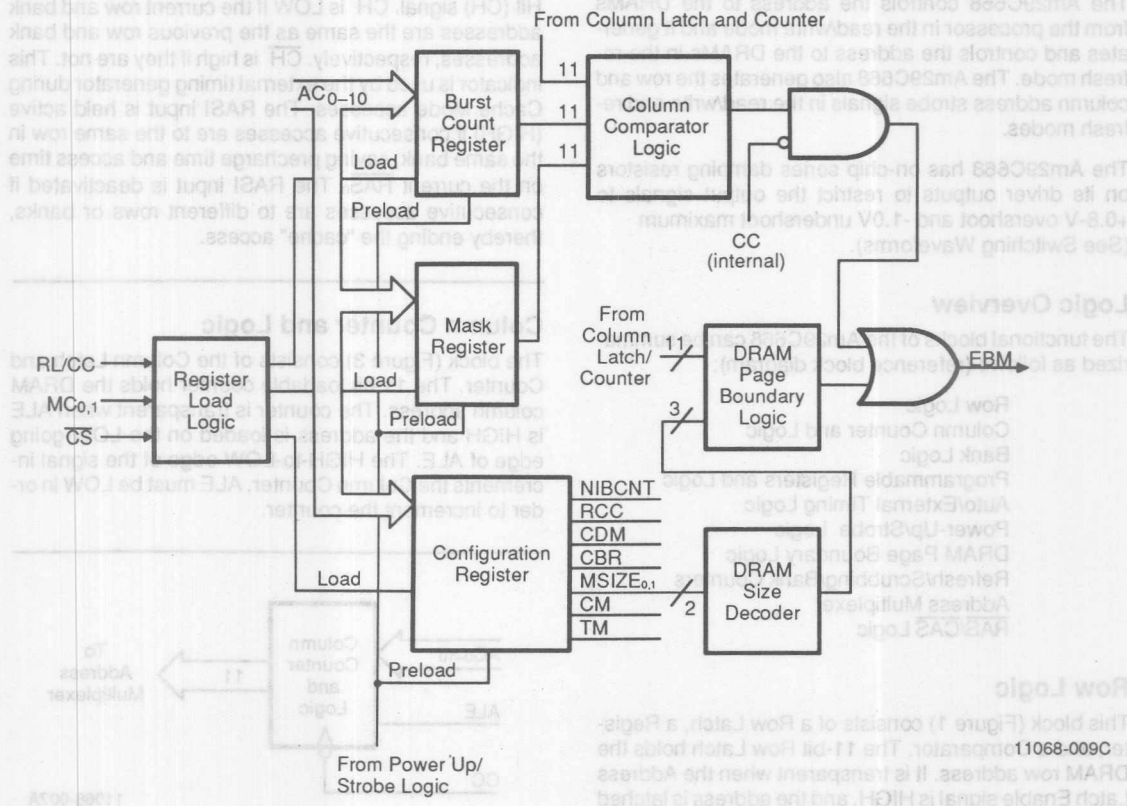


Figure 5. Programmable Registers and Logic

The 11-bit Burst Count Register is loaded via AC_{0-10} by the HIGH-going edge of RL through the Register Load Logic. This register is preloaded with all 1's (for a maximum burst count) in the Reset mode after power-up, and is only used in the Burst/Block mode of access, if a programmed number of accesses is required. This register is loaded with the maximum number of transfers to occur during any burst/block access. This number is dependent on the specifics of the system (i.e., page size or processor type).

The 11-bit Mask Register is loaded via AC_{0-10} by the HIGH-going edge of the RL signal through the Register Load Logic. This register is preloaded with all 1's (for all bits to be compared) in the Reset Mode after power-up, and is only used in the Burst/Block mode of access, if a programmed number of accesses is required.

This register allows a burst to be made which is shorter than the page size of the memory. When the Mask Register is loaded with a "1" in a bit location, the corresponding bit in the Column Latch and the Burst Count Register is compared. This register also aligns the Column Latch and Counter for succeeding bursts of full length if they occur immediately after termination of the prior burst. The Mask Register is loaded with 000000011111 for a 16-bit maximum burst.

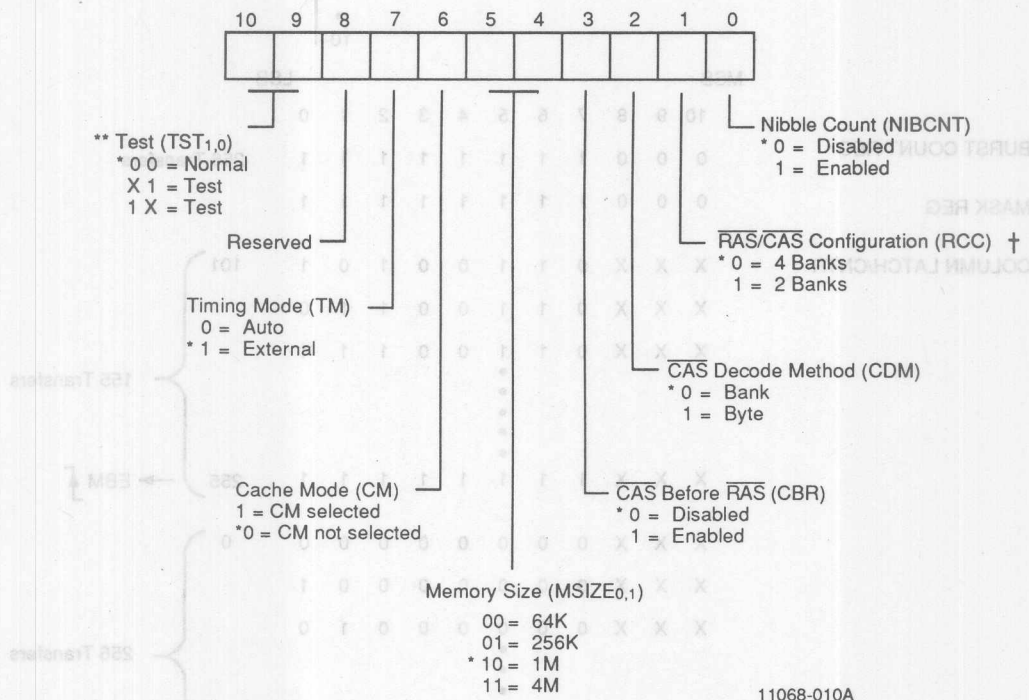
The Column Comparator Logic compares the contents of the Column Latch and Counter with that of the Burst Count Register (which contains the end of burst count value). The HIGH bits in the Mask Register determine which of the 11-bits of the Column Latch and Counter and the Burst Count Register are compared. This logic

is used only in the Burst/Block mode of access. Reference Figure 6a.

The Register Load Logic loads the Burst Count, Mask and Configuration Registers via the address bus (AC₀₋₁₀) dependent upon the state of the Register Load-

ing diagram in Figure 7. RL/CC Decoder and Register Load Logic are shown in Figure 8.

The DRAM Size Decoder determines the DRAM size being used. The MSIZE_{1,0} bits in the Configuration Register are used to decode the size of the DRAMs being used as shown in Figure 6.



* Default. The Am29C668 will power up in Am29368 mode if the user does not reprogram the configuration register.

† Reference Table 4.

** These bits are used during factory testing only.

Figure 6. Configuration Register Options

In the Auto Timing mode the CAS/CASIN input delays are generated from the active (HIGH) edge of RAS and is deasserted when RAS goes inactive by the Auto Timing Circuit. This internally generated CAS is gated with the CASIN input to generate the CAS output. This gating circuit allows the Auto Timing to be externally overridden (Figure 3). It is used for specially DRAM accesses.

In the External Timing mode (TM = 1), the internal CAS signal follows the externally generated CASIN input.

In this example, the self alignment feature of the Am29C668 is shown. The first burst is terminated on the DRAM page boundary by the EBM output. All subsequent bursts are then set at 256 transfers, which has been programmed via the Burst Count and Mask registers.

Auto/External Timing Logic

When Auto Timing mode is selected via the Timing Mode (TM) bit in the Configuration Register (TM = 0), the circuit generates internal timing delays between RAS-MSEL and MSEL-CAS. These delays are optimized for use with 100ns DRAMs.

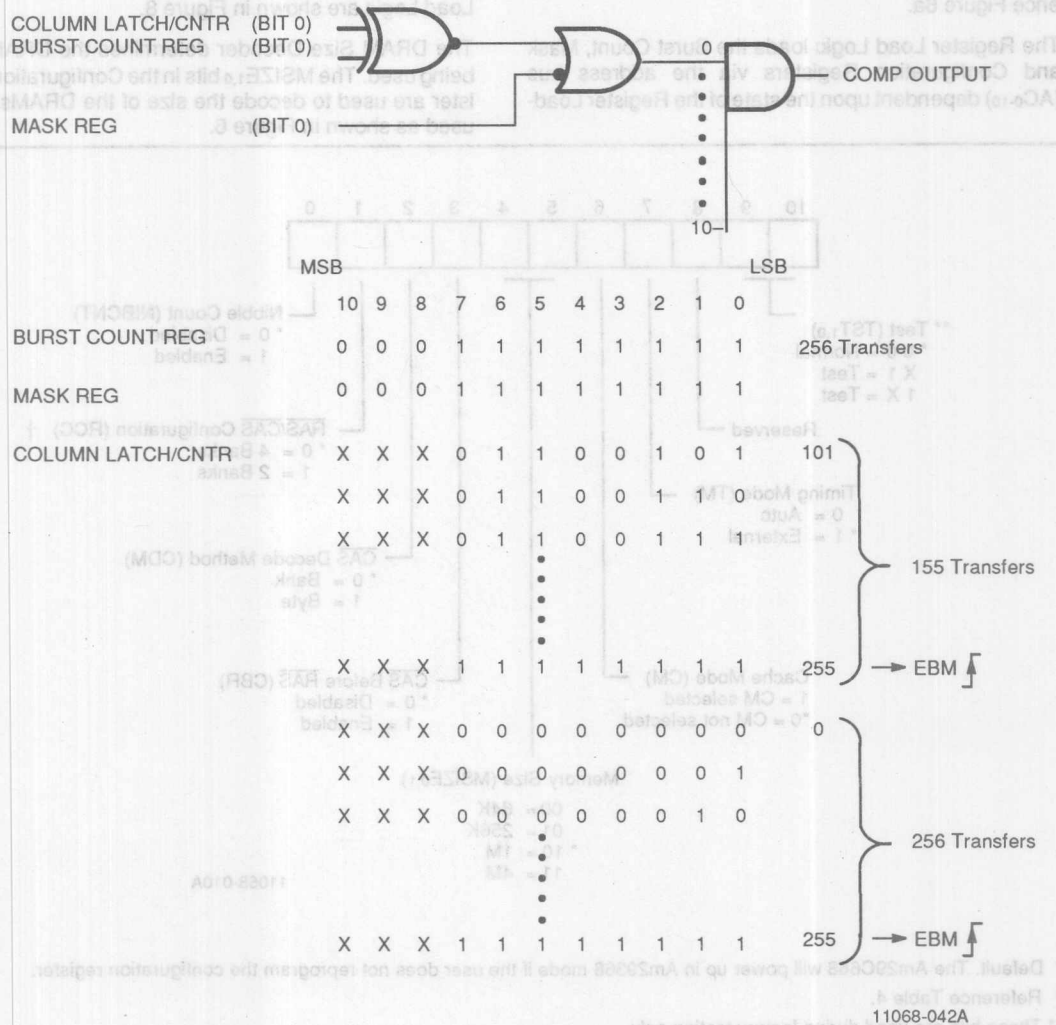


Figure 6a. Programmable Burst Logic with 256-word Burst Length Example

In this example, the self alignment feature of the Am29C668 is shown. The first burst is terminated on the DRAM page boundary by the EBM output. All subsequent bursts are then set at 256 transfers, which has been programmed via the Burst Count and Mask registers.

Auto/External Timing Logic

When Auto Timing mode is selected via the Timing Mode (TM) bit in the Configuration Register (TM = 0), this circuit generates internal timing delays between RASI-MSEL and MSEL-CASI. These delays are optimized for use with 100ns DRAMs.

In the Auto Timing mode the CASI/CASIEN input acts as CASIEN. In this mode internal $\overline{\text{CAS}}$ is generated from the active (HIGH) edge of RASI and is deactivated when RASI goes inactive by the Auto Timing Circuit. This internally generated $\overline{\text{CAS}}$ is gated with the CASIEN input to generate the $\overline{\text{CAS}}$ outputs. This gating circuit allows the Auto-Timing to be externally overridden (Figure 9). It is used for specialty DRAM accesses.

In the External Timing mode (TM = 1), the internal $\overline{\text{CAS}}$ signal follows the externally generated CASI input.

In the Auto Timing Mode the MSEL/MSELEN input acts as MSELEN. In this mode, internal MSEL is generated from the active (HIGH) edge of RAS1, and is deactivated when RAS1 goes inactive, by the Auto Timing Circuit. This internally generated MSEL is gated with the MSELEN input to generate the internal MSEL signal. This feature is used to extend row address hold time via external control (overriding the Auto Timing feature).

In the External Timing mode (TM = 1), the internal MSEL signal follows the externally generated MSEL input.

The Auto Timing mode allows 4 banks of 16-bit data plus 6 EDC check bits or 2 banks of 32-bit data plus 7 EDC

check bits comprised of 100ns DRAMs to be operated without external drivers.

Power-up/Strobe Logic

This block automatically presets the Am29C668 to the default condition upon power-up (Figure 6). This circuit also generates all the internal control signals for the Refresh Counter, Configuration, Burst Count, and Mask Registers, the Register Load Logic, and the Bank Register.

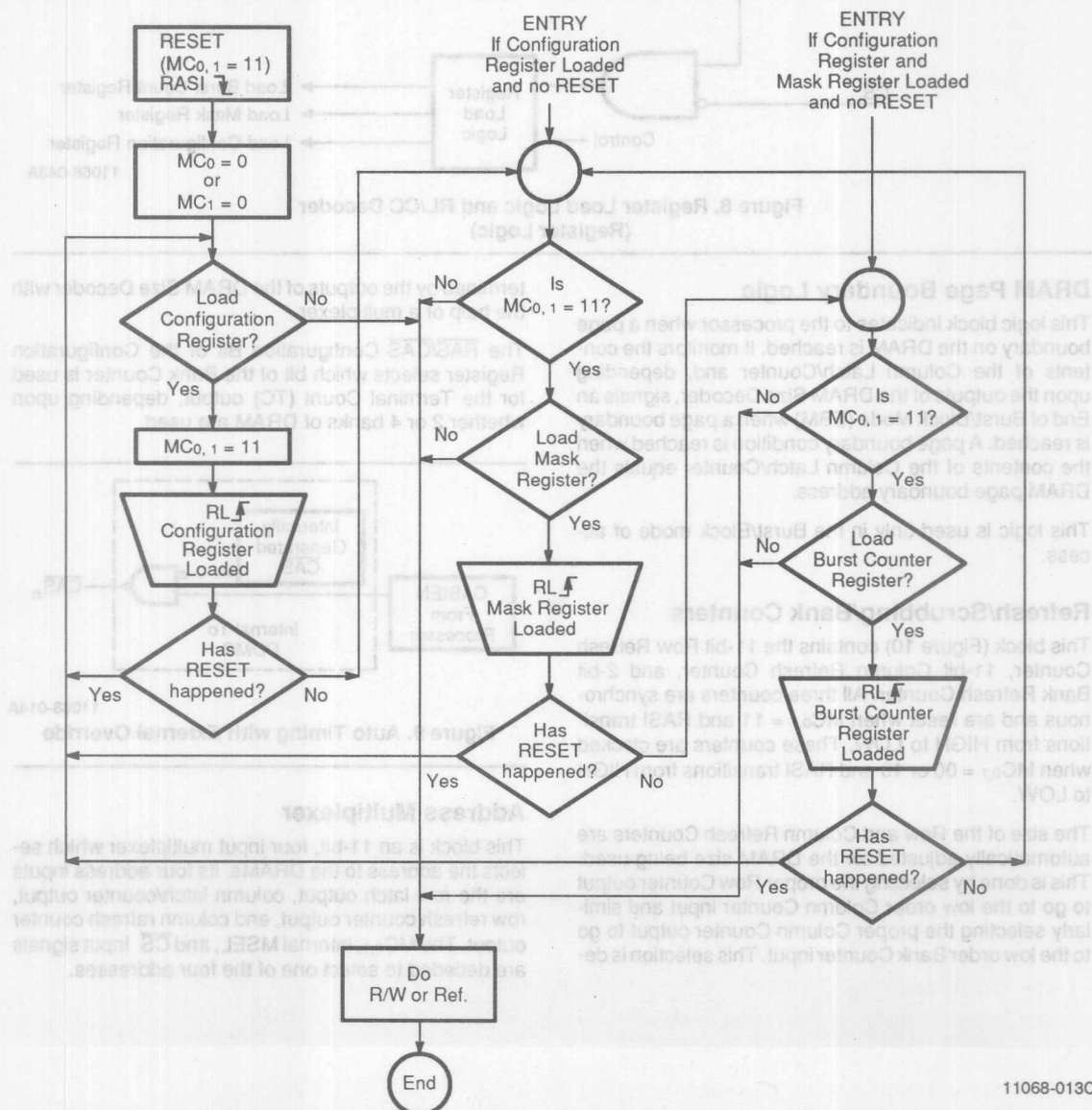


Figure 7. Register Loading (See paragraph on next page)

11068-013C

Figure 7. Register Loading. The Configuration Register must be loaded before the Mask and Burst Count Registers may be loaded. Once the Configuration Register is loaded, the Register Load Logic will toggle between loading the Mask Register and Burst Count Register. The Configuration Register may only be loaded

immediately after a reset. The Mask Register and Burst Count Register are only used in the Burst/Block access mode, in other modes only the Configuration Register need be loaded if the user wishes to alter its default mode indicated in Figure 6.

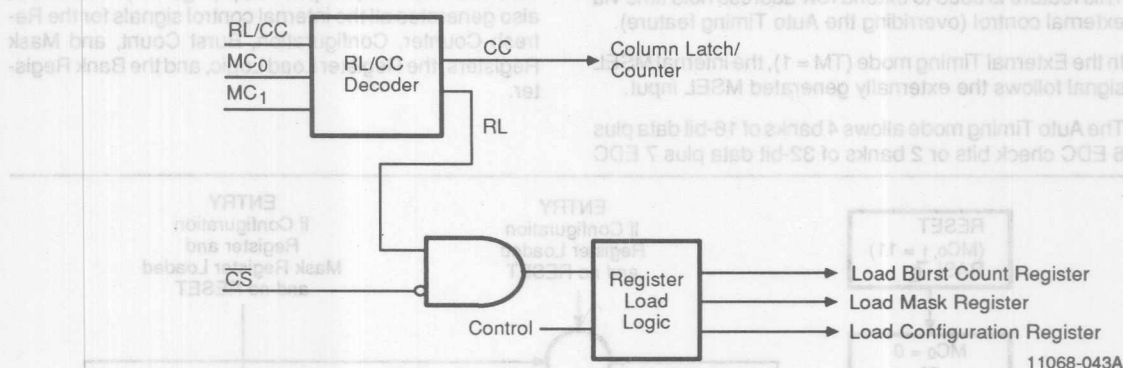


Figure 8. Register Load Logic and RL/CC Decoder (Register Logic)

DRAM Page Boundary Logic

This logic block indicates to the processor when a page boundary on the DRAM is reached. It monitors the contents of the Column Latch/Counter and, depending upon the outputs of the DRAM Size Decoder, signals an End of Burst/Block Mode (EBM) when a page boundary is reached. A page boundary condition is reached when the contents of the Column Latch/Counter equals the DRAM page boundary address.

This logic is used only in the Burst/Block mode of access.

terminated by the outputs of the DRAM Size Decoder with the help of a multiplexer.

The RAS/CAS Configuration Bit of the Configuration Register selects which bit of the Bank Counter is used for the Terminal Count (TC) output, depending upon whether 2 or 4 banks of DRAM are used.

Refresh/Scrubbing/Bank Counters

This block (Figure 10) contains the 11-bit Row Refresh Counter, 11-bit Column Refresh Counter, and 2-bit Bank Refresh Counter. All three counters are synchronous and are reset when $MC_{0,1} = 11$ and RAS transitions from HIGH to LOW. These counters are clocked when $MC_{0,1} = 00$ or 10 and RAS transitions from HIGH to LOW.

The size of the Row and Column Refresh Counters are automatically adjusted for the DRAM size being used. This is done by selecting the proper Row Counter output to go to the low order Column Counter input and similarly selecting the proper Column Counter output to go to the low order Bank Counter input. This selection is de-

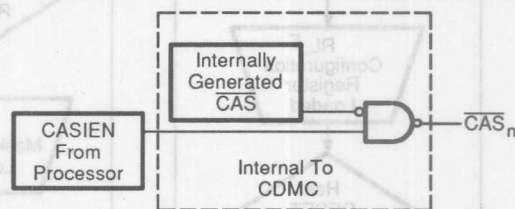


Figure 9. Auto Timing with External Override

Address Multiplexer

This block is an 11-bit, four input multiplexer which selects the address to the DRAMs. Its four address inputs are the row latch output, column latch/counter output, row refresh counter output, and column refresh counter output. The $MC_{0,1}$, internal MSEL, and \overline{CS} input signals are decoded to select one of the four addresses.

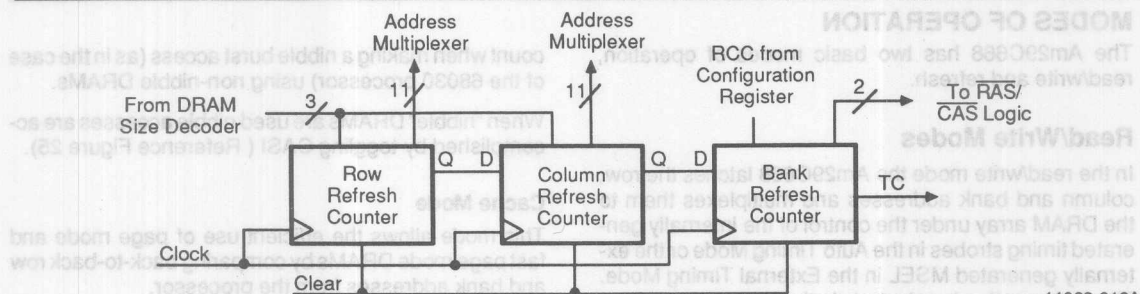


Figure 10. Refresh/Scrubbing/Bank Counters

RAS/CAS Logic

This block (Figure 11) contains the $\overline{\text{CAS}}$ Before $\overline{\text{RAS}}$ Logic, $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$ Decode Logic, $\overline{\text{CAS}}$ Enable Logic, and $\overline{\text{CAS}}$ Multiplexer.

The $\overline{\text{CAS}}$ Before $\overline{\text{RAS}}$ Logic switches the $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$ lines to the $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$ Decode Logic if the CBR selection bit in the Configuration Register is set (1). This allows a $\overline{\text{CAS}}$ Before $\overline{\text{RAS}}$ refresh to be accomplished without altering the order of the RAS and CAS input strobes. Refresh With Scrubbing ($\text{MC}_{0,1} = 10$) is not allowed when the CBR bit is set (1).

The $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$ Decode Logic decodes the internal $\overline{\text{RAS}}$ and $\overline{\text{CAS}}$ timing signals to generate the four $\overline{\text{RAS}}_n$ and four $\overline{\text{CAS}}_n$ output signals.

The $\overline{\text{CAS}}$ Enable Logic is used if a byte-wise $\overline{\text{CAS}}$ decoding method is selected. Byte enables are decoded externally and are connected to the $\overline{\text{CAS}}_{\text{EN}0-3}$ inputs. In the $\overline{\text{CAS}}$ Enable Logic all the $\overline{\text{CAS}}_{\text{EN}0-3}$ signals are individually gated with the internal $\overline{\text{CAS}}$ signal to generate the $\overline{\text{CAS}}_n$ proper outputs.

The $\overline{\text{CAS}}$ Multiplexer is a 4-bit, two input multiplexer. It selects one set of $\overline{\text{CAS}}_n$ signals to the output depending upon the $\overline{\text{CAS}}$ decode method being used (selected by the CDM bit in the Configuration Register) and the operating mode (selected by $\text{MC}_{0,1}$).

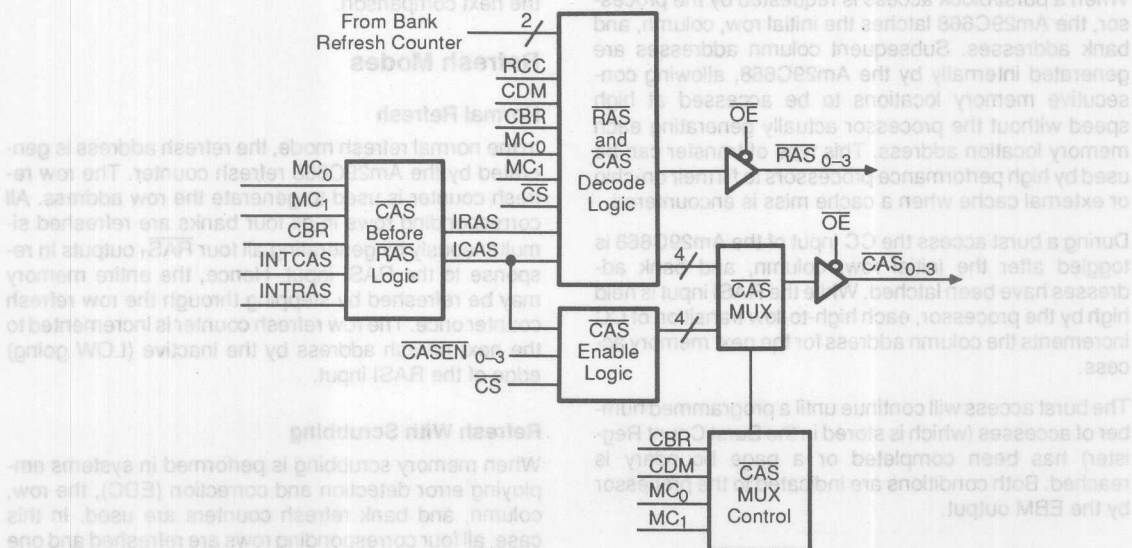


Figure 11. RAS/CAS Logic

MODES OF OPERATION

The Am29C668 has two basic modes of operation, read/write and refresh.

Read/Write Modes

In the read/write mode the Am29C668 latches the row, column and bank addresses and multiplexes them to the DRAM array under the control of the internally generated timing strobes in the Auto Timing Mode or the externally generated MSEL in the External Timing Mode. The timing option is selected via the Timing Mode (TM) bit in the Configuration Register (Figure 6).

The row address is latched in the DRAMs by the active (LOW-going) edge of the \overline{RAS}_n output, which follows the active (HIGH-going) edge of the RAS_I input. The address lines are then switched to column address by either an internally generated signal in the Auto Timing Mode or by pulling MSEL active HIGH in the External Timing Mode. The column address is latched in the DRAMs on the active (LOW-going) edge of the \overline{CAS}_n output, which follows either an internally generated signal in the Auto Timing Mode or the active (HIGH-going) edge of the CAS_I input in the External Timing Mode.

The read/write mode of the Am29C668 may be optimized for the shortest memory cycle time, through burst/block accesses, nibble mode accesses, or "cache" mode accesses.

Burst/Block Mode

When a burst/block access is requested by the processor, the Am29C668 latches the initial row, column, and bank addresses. Subsequent column addresses are generated internally by the Am29C668, allowing consecutive memory locations to be accessed at high speed without the processor actually generating each memory location address. This type of transfer can be used by high performance processors to fill their on-chip or external cache when a cache miss is encountered.

During a burst access the CC input of the Am29C668 is toggled after the initial row, column, and bank addresses have been latched. While the RAS_I input is held high by the processor, each high-to-low transition of CC increments the column address for the next memory access.

The burst access will continue until a programmed number of accesses (which is stored in the Burst Count Register) has been completed or a page boundary is reached. Both conditions are indicated to the processor by the EBM output.

Nibble Mode

For Nibble mode accesses the Nibble Count bit (NIBCNT) in the Configuration Register is set to "1". This bit enables only the two least significant bits of the Column Latch and Counter to be clocked, allowing the Column Latch and Counter to perform a modulo four

count when making a nibble burst access (as in the case of the 68030 processor) using non-nibble DRAMs.

When "nibble" DRAMs are used nibble accesses are accomplished by toggling CAS_I (Reference Figure 25).

Cache Mode

This mode allows the efficient use of page mode and fast page mode DRAMs by comparing back-to-back row and bank addresses from the processor.

In the "cache" mode of access of the Am29C668 the \overline{RAS}_n output is held active (LOW) and any location in that row is accessed by only changing the column address. This makes the entire row look like a cache, since any access in that row can be made at high speed. To select the cache access mode, the Cache Mode (CM) bit in the Configuration Register is set to "1". The row and bank addresses of consecutive accesses are compared by the Am29C668. If the row and bank addresses of consecutive accesses match, CH goes active (LOW) and signals the timing generator not to deactivate the RAS_I input but only to toggle the CAS_I/CAS_{IEN} input. If the row and bank addresses of consecutive accesses do not match, the CH signal goes inactive (HIGH) and informs the timing generator to deactivate the RAS_I input and start a new RAS_I cycle after the current cycle has gone through a \overline{RAS} precharge cycle. When the RAS_I input is activated, its HIGH-going edge loads the row and bank registers with the contents of the row and bank latches, respectively, saving the current values for the next comparison.

Refresh Modes

Normal Refresh

In the normal refresh mode, the refresh address is generated by the Am29C668 refresh counter. The row refresh counter is used to generate the row address. All corresponding rows in all four banks are refreshed simultaneously by generating all four \overline{RAS}_n outputs in response to the RAS_I input. Hence, the entire memory may be refreshed by stepping through the row refresh counter once. The row refresh counter is incremented to the next refresh address by the inactive (LOW going) edge of the RAS_I input.

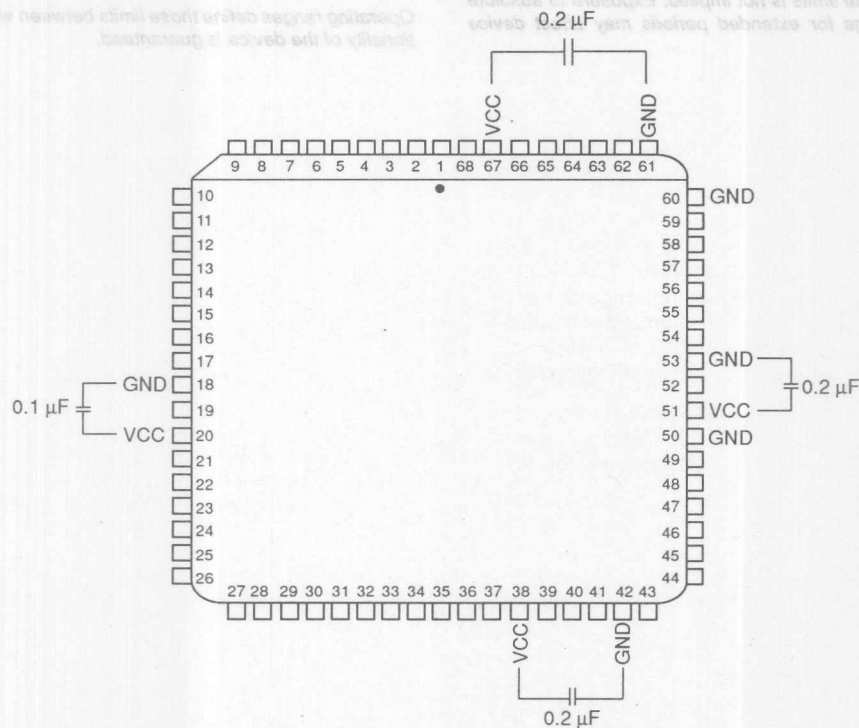
Refresh With Scrubbing

When memory scrubbing is performed in systems employing error detection and correction (EDC), the row, column, and bank refresh counters are used. In this case, all four corresponding rows are refreshed and one location of one row is "scrubbed", i.e., a read/modify/write cycle is performed. An entire memory array can be scrubbed by stepping through the row, column, and bank address counters once. The Am29C668 has four independent \overline{CAS}_n outputs allowing a single bit to be accessed during refresh cycles.

CAS Before RAS Refresh

This is a feature of some dynamic RAMs. The DRAM on-chip refresh counter is updated and a refresh cycle performed by generating a $\overline{\text{CAS}}$ strobe before the $\overline{\text{RAS}}$ strobe. This refresh support is selected by enabling (setting to "1") the $\overline{\text{CAS}}$ Before $\overline{\text{RAS}}$ bit in the Configuration Register. Refresh with scrubbing ($\text{MC}_{0,1} = 10$) is not allowed when the CBR bit is set (1).

In this mode, $\overline{\text{RAS}}$ controls the $\overline{\text{CAS}}$ outputs and CAS controls the $\overline{\text{RAS}}$ outputs. This allows a $\overline{\text{CAS}}$ before $\overline{\text{RAS}}$ refresh to be accomplished without altering the order of the $\overline{\text{RAS}}$ and CAS input strobes. When this mode is set, memory "scrubbing" ($\text{MC}_{0,1} = 10$) is prohibited.



11068-042A

Figure 12. Device Decoupling – Vcc and Ground Pin Connections

Notes:

Due to the high switching speeds and high drive capability of the Am29C668, it is necessary to decouple the device for proper operation. Multilayer ceramic capacitors are recommended. It is important to mount the capacitors as close as possible to the power pins (Vcc, GND) to minimize lead inductance and noise. A ground plane is strongly recommended. A wire wrapped board without power and ground planes is not recommended.

It is strongly recommended that this part be directly surface mounted whenever possible. Should a PLCC, or PGA socket be required, a one-time-insertion-only socket with minimal lead length is necessary for proper device function. The socket lead inductance should be 8nH maximum per pin.

The socket may be obtained from Methode Electronics (Part #213-068-101) at (800) 323-6858 or AMP (Part #821574-1, 821574-3, 641749-2) at (800) 522-6752.

ABSOLUTE MAXIMUM RATINGS

Storage Temperature	-65 to +150°C
Ambient Operating Temperature	-55 to +125°C
Maximum Vcc	-0.5 to +7.0 V
DC Voltage Applied to Any Pin	-0.5 to (Vcc + 0.3) V

Stresses above those listed under ABSOLUTE MAXIMUM RATINGS may cause permanent device failure. Functionality at or above these limits is not implied. Exposure to absolute maximum ratings for extended periods may affect device reliability.

OPERATING RANGES

Commercial (C) Devices

Ambient Temperature (Ta)	0 to +70°C
Supply Voltage (Vcc)	+4.5 to +5.5 V

Military (M)

Case Temperature (Tc)	-55 to +125°C
Supply Voltage (Vcc)	+4.5 to +5.5 V

Operating ranges define those limits between which the functionality of the device is guaranteed.

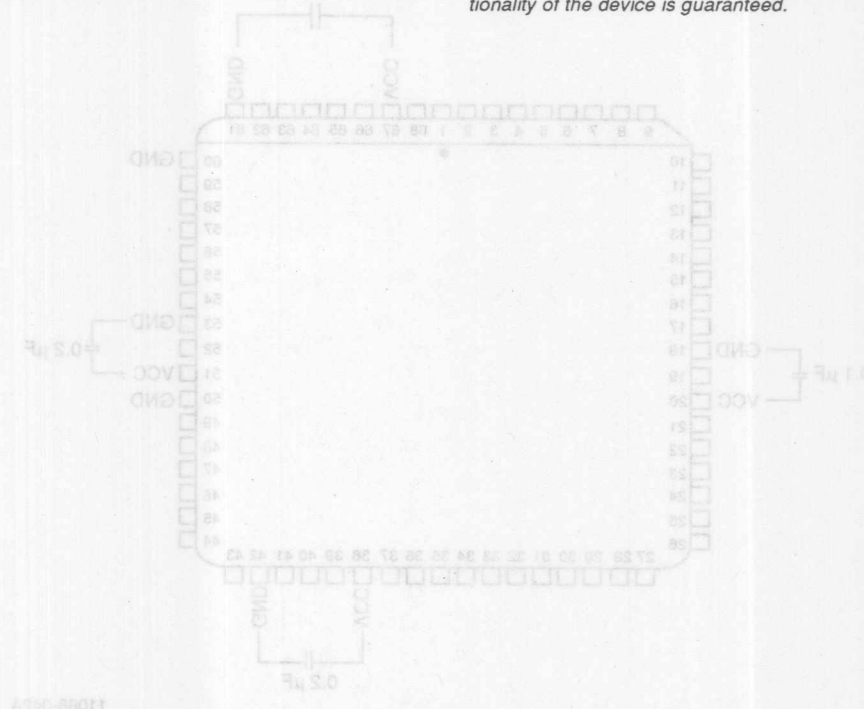





Figure 15. Device Decoupling - Vcc and Ground Pin Connections

Notes:
 Due to the high switching speeds and high drive capability of the Am29C668, it is necessary to decouple the device for proper operation. Multilayer ceramic capacitors are recommended. It is important to mount the capacitors as close as possible to the power pins (Vcc, GND) to minimize lead inductance and noise. A ground plane is strongly recommended. A wire-wound board without power and ground planes is not recommended.
 It is strongly recommended that the part be directly surface mounted whenever possible. Should a PLCC or PGA socket be required, a one-time-insertion-only socket with minimal lead length is necessary for proper device function. The socket lead inductance should be less than maximum per pin.
 The socket may be obtained from Multichip Electronics (Part #S10-068-101) at (800) 328-6828 or AMP (Part #621274-1, 621274-2, 641248-2) at (800) 522-6725.

DC CHARACTERISTICS over operating range unless otherwise specified (for APL Products, Group A, Subgroups 1,2,3 are tested unless otherwise noted)

Parameter Symbol	Parameter Description	Test Conditions (Note 1)	Min.	Max.	Unit
V_{IH}	Input HIGH Voltage	Guaranteed Input Logical HIGH Voltage for all Inputs (Note 2)	2.0		V
V_{IL}	Input LOW Voltage	Guaranteed Input Logical LOW Voltage for all Inputs (Note 2)		0.8	V
I_{IH}	Input HIGH Current	$V_{CC} = \text{Max.}, V_{IN} = V_{CC}$		5.0	μA
I_{IL}	Input LOW Current	$V_{CC} = \text{Max.}, V_{IN} = \text{GND}$		-5.0	μA
V_{OH}	Output HIGH Voltage	$V_{CC} = \text{Min.}, I_{OH} = -10 \text{ mA}$	2.7		V
V_{OL}	Output LOW Voltage	$V_{CC} = \text{Min.}, I_{OL} = 10 \text{ mA}$		0.5	V
V_{ON}	Output Undershoot  Voltage (Note 4)	$C_L = 50 \text{ pF}$		-1.0	V
V_{OP}	Output Overshoot  Voltage (Note 4)	$C_L = 50 \text{ pF}$		0.8	V
V_{OP}	Output Overshoot  Voltage (Note 4)	$C_L = 50 \text{ pF}$		0.8	V
I_{OZ}	Off-State (High Impedence) Output Current; Q Outputs	$V_{CC} = \text{Max.}, V_O = 0 \text{ V}$		-10	μA
		$V_O = V_{CC}(\text{Max})$		10	
I_{CCQ}	Quiescent Power Supply Current (CMOS Inputs)	$V_{CC} = \text{Max.}, 4.3 \text{ V} \leq V_{IN}, V_{IN} \leq 0.2 \text{ V}, f_{OP} = 0$		5.0	mA
I_{CCT}	Quiescent Input Power Supply Current (@ TTL HIGH)	$V_{CC} = \text{Max.}, V_{IN} = 2.4, f_{OP} = 0$		25	mA
I_{CCD}	Dynamic Power Supply Current (Note 5)	$V_{CC} = \text{Max.}, 4.3 \text{ V} \leq V_{IN}, V_{IN} \leq 0.2 \text{ V}, C_L = 150 \text{ pF}, \text{OE} = \text{LOW}$	MIL	7	mA/MHz
			COM'L	7	
I_{CC}	Total Power Supply Current (Notes 3 and 5)	$V_{CC} = \text{Max.}, f_{OP} = 10 \text{ MHz}, \text{OE} = \text{LOW}, 50\% \text{ Duty cycle}, C_L = 150 \text{ pF}, 4.3 \text{ V} \leq V_{IN}, V_{IN} \leq 0.2 \text{ V}$	MIL	100	mA
			COM'L	100	
		$V_{CC} = \text{Max.}, f_{OP} = 10 \text{ MHz}, \text{OE} = \text{LOW}, 50\% \text{ Duty cycle}, C_L = 150 \text{ pF}, V_{IN} = 3.4, V_{IL} = 0.4 \text{ V}$	MIL	100	
			COM'L	100	

Notes:

- For conditions shown as Min. or Max., use appropriate value specified under Operating Range for the applicable device type.
- Tested with limited test pattern.
- Total Power Supply Current is the sum of the Quiescent Current and the dynamic current (at either CMOS or TTL input levels). For all conditions, the Total Power Supply Current can be calculated by using the following equation:

$$I_{CC} = I_{CCQ} + I_{CCD}(f_{OP}) \text{ (CMOS Inputs), } I_{CC} = I_{CCT} + I_{CCD}(f_{OP}) \text{ (TTL Inputs), } f_{OP} = \text{Operating Frequency in Megahertz}$$

During device characterization, two addresses, one RAS_n and one CAS_n output were toggled at $f_{OP} = 10 \text{ MHz}$ during I_{CC} measurement
- V_{ON} and V_{OP} are not production tested but are guaranteed by characterization data for surface mounted devices with proper capacitive decoupling. Limits specified are for all outputs switching simultaneously with minimum specified loading. As loading increases, V_{ON} and V_{OP} will approach zero. Reference Switching Waveforms.
- Not tested in production. Guaranteed by characterization data.

SWITCHING CHARACTERISTICS over operating range unless otherwise specified
 Capacitive Loading = 350 pF for all Q_n, $\overline{\text{RAS}}_n$, and $\overline{\text{CAS}}_n$; 150 pF for all other outputs (minimum tester load). See Note 1.

No.	Parameter Symbol	Parameter Description	Test Conditions	Commercial				Military		Unit
				Am29C668		Am29C668-1		Am29C668-1		
				Min.	Max.	Min.	Max.	Min.	Max.	
COMMON PARAMETERS										
1	t _{PD}	ACn/ARn to Qn (Note 4)	ALE = 1	2	34	3	29	0	35	ns
2	t _{PD}	MCn to Qn (Note 4)		6	35	7	30	4	37	ns
3	t _{PD}	ALE to Qn (Note 4)		4	34	5	29	3	37	ns
4	t _{PD}	$\overline{\text{CS}}$ to Qn (Note 4)		6	36	7	32	6	39	ns
5	t _s	ACn/ARn to ALE \downarrow Set up time		4		4		3		ns
6	t _H	ACn/ARn to ALE \downarrow Hold time		5		5		4		ns
7	t _s	SELn to ALE \downarrow Set up time		4		4		3		ns
8	t _H	SELn to ALE \downarrow Hold time		4		4		3		ns
9	t _s	MCn to RAS \uparrow Set up time		5		5		3		ns
10	t _H	MCn to RAS \downarrow Hold time		5		5		3		ns
10A	t _s	MCn to RAS \downarrow Set up time	MCn = 11	4		4		3		ns
11	t _s	$\overline{\text{CS}} \downarrow$ to RAS \uparrow	(Note 3)	4		4		3		ns
12	t _H	$\overline{\text{CS}} \uparrow$ to RAS \downarrow	(Note 3)	4		4		3		ns
13	t _s	SELn to RAS \uparrow Set up time	ALE = 1	4		4		3		ns
14	t _H	SELn to RAS \downarrow Hold time	ALE = 1	4		4		3		ns
15	t _{PWL}	RAS Pulse Width LOW		10		10		10		ns
16	t _{PWH}	RAS Pulse Width HIGH		10		10		10		ns
EXTERNAL TIMING MODE										
17	t _{PD}	RAS \uparrow to $\overline{\text{RAS}}_n \downarrow$ (active edges) (Note 4)		9	30	9	26	8	32	ns
17A	t _{PD}	RAS \downarrow to $\overline{\text{RAS}}_n \uparrow$ (inactive edges) (Note 4)		4	23	4	21	3	23	ns
18	t _{PD}	CAS \uparrow to $\overline{\text{CAS}}_n \downarrow$ (active edges) (Note 4)		7	31	8	27	9	32	ns
18A	t _{PD}	CAS \downarrow to $\overline{\text{CAS}}_n \uparrow$ (inactive edges) (Note 4)		4	23	5	21	4	23	ns
19	t _{PD}	MSEL to Qn		4	34	8	30	3	37	ns
AUTO TIMING MODE										
20	t _{PD}	RAS \uparrow to $\overline{\text{RAS}}_n \downarrow$ (active edges) (Note 4)		9	30	9	26	8	32	ns
20A	t _{PD}	RAS \downarrow to $\overline{\text{RAS}}_n \uparrow$ (inactive edges) (Note 4)		4	23	4	20	3	23	ns
21	t _{SKEW}	$\overline{\text{RAS}}_n$ to Qn (row address) (Note 4)	MSELEN = 1	12		13		9		ns
21A	t _{PD}	RAS to Qn (column address)	MSELEN = 1	20	70	20	65	20	75	ns
22	t _{PD}	RAS \uparrow to $\overline{\text{CAS}}_n \downarrow$	CASIEN = 1 (Note 2)		98		88		102	ns
23	t _{SKEW}	$\overline{\text{RAS}}_n$ to $\overline{\text{CAS}}_n \downarrow$	CASIEN = 1	22	78	25	72	20	82	ns
24	t _{SKEW}	Qn (column address) to $\overline{\text{CAS}}_n$	MSELEN = 1 CASIEN = 1	4		4		4		ns

SWITCHING CHARACTERISTICS (Continued)

No.	Parameter Symbol	Parameter Description	Test Conditions	Commercial				Military		Unit
				Am29C668		Am29C668-1		Am29C668-1		
				Min.	Max.	Min.	Max.	Min.	Max.	
AUTO TIMING MODE										
25	t _{PD}	MSELEN to Qn (column address) (Note 4)		6	34	7	30	6	37	ns
26	t _{PD}	CASIEN _↑ to \overline{CASn} _↓ (active edges) (Note 4)		6	31	7	27	8	32	ns
26A	t _{PD}	CASIEN _↓ to \overline{CASn} _↑ (inactive edges) (Note 4)		3	23	4	21	4	23	ns
SPECIALTY MODES										
27	t _{PD}	CC _↓ to Qn (Note 4)		5	32	6	29	6	34	ns
28	t _{PD}	CC _↓ to EBM		4	23	5	20	4	25	ns
29	t _{PW}	CC Pulse Width LOW or HIGH		10		10		12		ns
32	t _{PD}	ARn to \overline{CH}	ALE = 1		19		17		21	ns
33	t _{PD}	ALE to EBM _↓		4	20	4	18	4	22	ns
34	t _{PD}	ALE to \overline{CH}			17		15		18	ns
35	t _{PD}	RASI _↓ to TC			30		24		35	ns
36	t _s	ACn to RL _↑ Set up time		3		2		0		ns
37	t _H	ACn to RL _↑ Hold time		12		11		12		ns
38	t _s	MCn, \overline{CS} to RL _↑ Set up time		7		6		7		ns
39	t _H	MCn, \overline{CS} to RL _↑ Hold time		7		6		7		ns
40	t _{PW}	RL Pulse Width LOW or HIGH		10		10		10		ns
41	t _{PD}	\overline{CASENn} to \overline{CASn} (Note 4)		2	27	3	23	3	30	ns
OUTPUT SKEWS (worst case for any given device)										
42	t _{SKEW}	{t _{PD} (ACn/ARn to Qn)–t _{PD} (RASI to \overline{RASn})}	MCn = 01		20		18		21	ns
42A	t _{SKEW}	{t _{PD} (ACn/ARn to Qn)–t _{PD} (CASI to \overline{CASn})}	MCn = 01		18		16		18	ns
43	t _{SKEW}	{t _{PD} (MCn to Qn)–t _{PD} (RASI to \overline{RASn})}	MCn = 00, 01		22		19		23	ns
44	t _{SKEW}	{t _{PD} (MSEL to Qn)–t _{PD} (RASI to \overline{RASn})}	MCn = 01		21		18		23	ns
45	t _{SKEW}	{t _{PD} (MSEL to Qn)–t _{PD} (CASI to \overline{CASn})}	MCn = 01		19		17		20	ns
THREE-STATE OUTPUTS										
46	t _{PLZ}	Output Disable Time from LOW	C _L = 350 pF		24		24		25	ns
47	t _{PHZ}	Output Disable Time from HIGH			28		26		30	ns
48	t _{PZL}	Output Enable Time from LOW			18		16		18	ns
49	t _{PZH}	Output Enable Time from HIGH			18		16		18	ns

Notes:

1. To calculate propagation delays at loads other than those specified, see section labeled "TYPICAL Change in Propagation Delay vs. Loading Capacitance" on the following page.
2. For deassertion, removal of RASi will deassert \overline{CASn} with t_{PD} indicated by external timing parameter 18A.
3. Not included in Group A testing. Not production tested. These parameters are included to ease system design—for reference only.
4. Refer to Figure 13 for information on how the min. and max. parameters are specified.

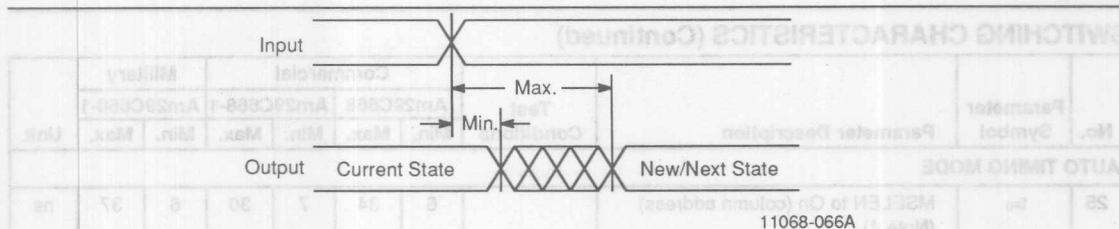
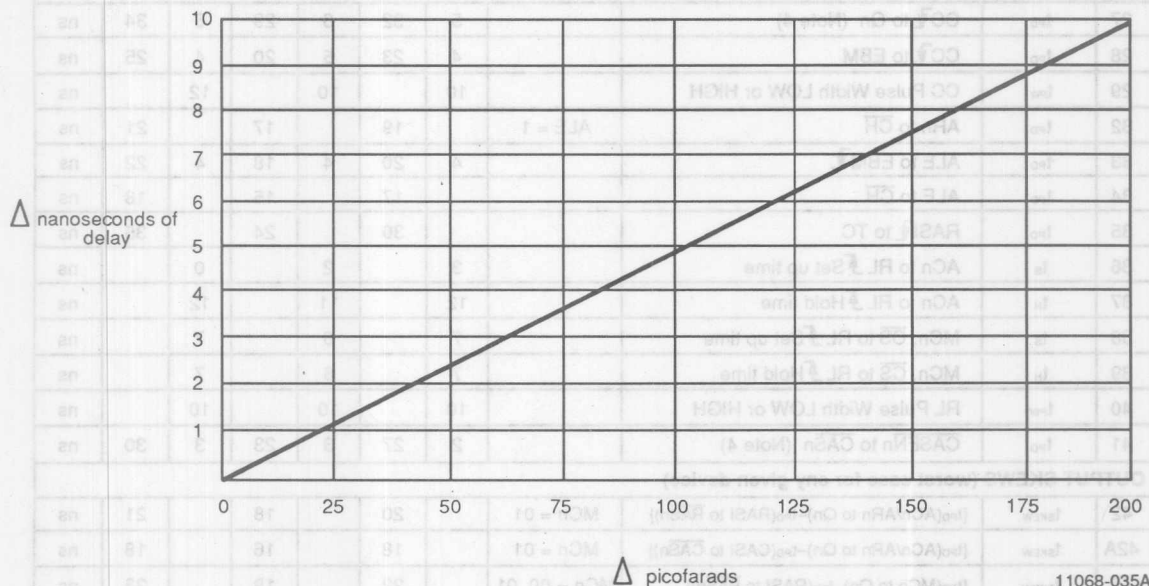


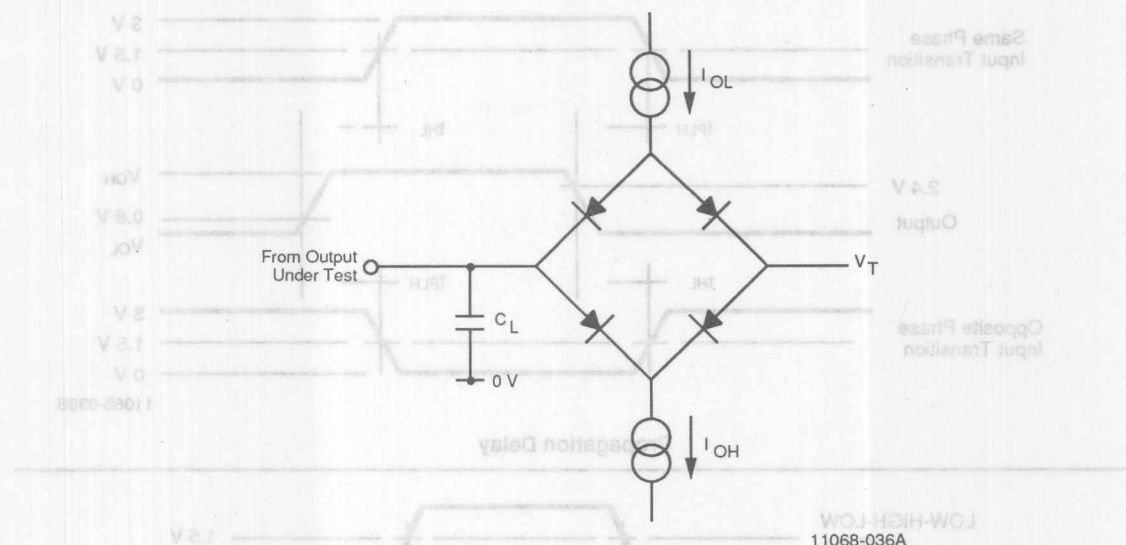
Figure 13. Minimum and Maximum Timings Specification

TYPICAL Change in Propagation Delay
vs Loading Capacitance



- Notes:
1. To calculate propagation delays at loads other than those specified, see section labeled "TYPICAL Change in Propagation Delay vs. Loading Capacitance" on the following page.
 2. For transition removal of RAS1 will be used with RAS2 indicated by external timing parameter 10A.
 3. Not included in Group A testing. Not production tested. These parameters are included to ease system design—for reference only.
 4. Refer to Figure 13 for information on how the min. and max. parameters are specified.

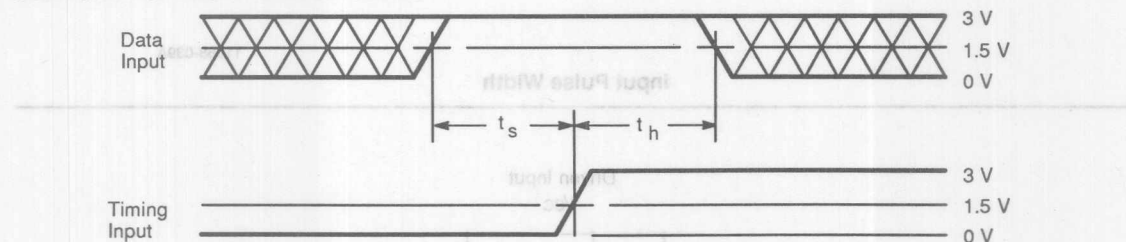
SWITCHING TEST CIRCUIT



Notes:

1. Diagram shown for HIGH data only. Output transition may be opposite sense.
2. Cross-hatched area is don't care condition.

SWITCHING TEST WAVEFORMS



Notes:

1. Diagram shown for HIGH data only. Output transition may be opposite sense.
2. Cross-hatched area is don't care condition.

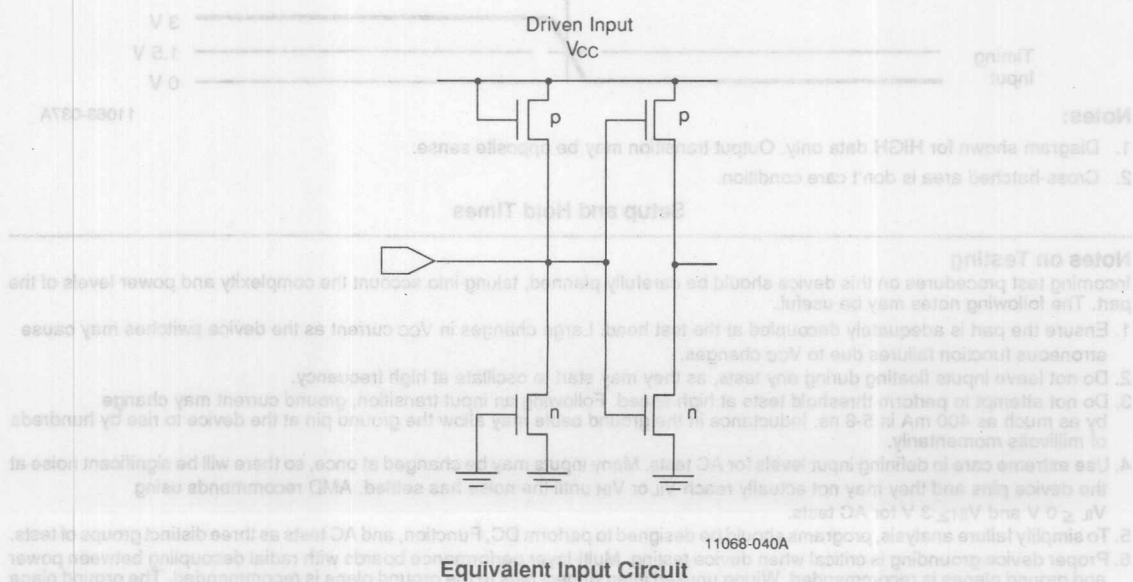
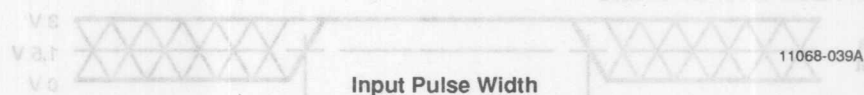
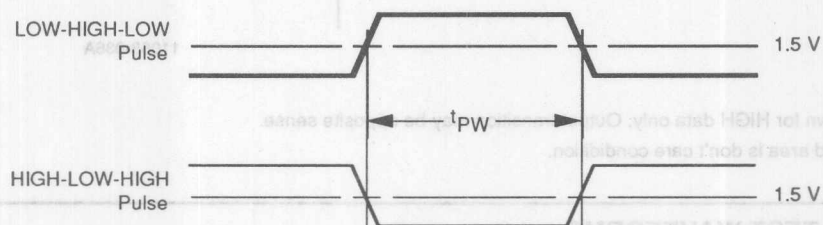
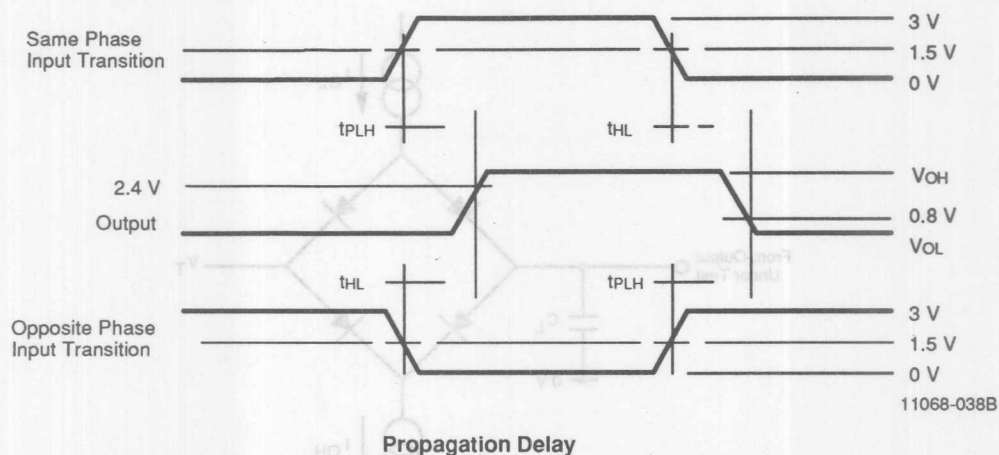
Setup and Hold Times

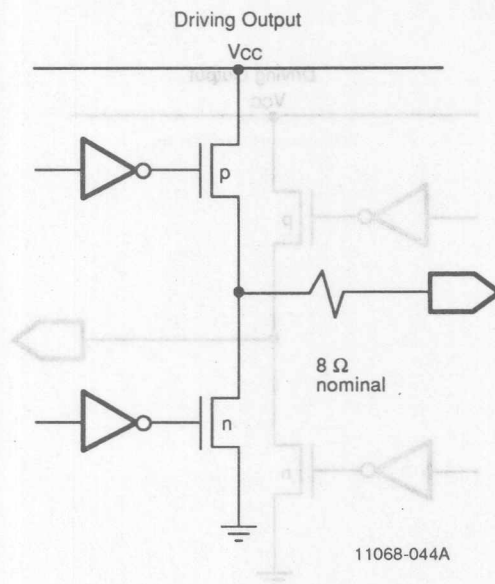
Notes on Testing

Incoming test procedures on this device should be carefully planned, taking into account the complexity and power levels of the part. The following notes may be useful.

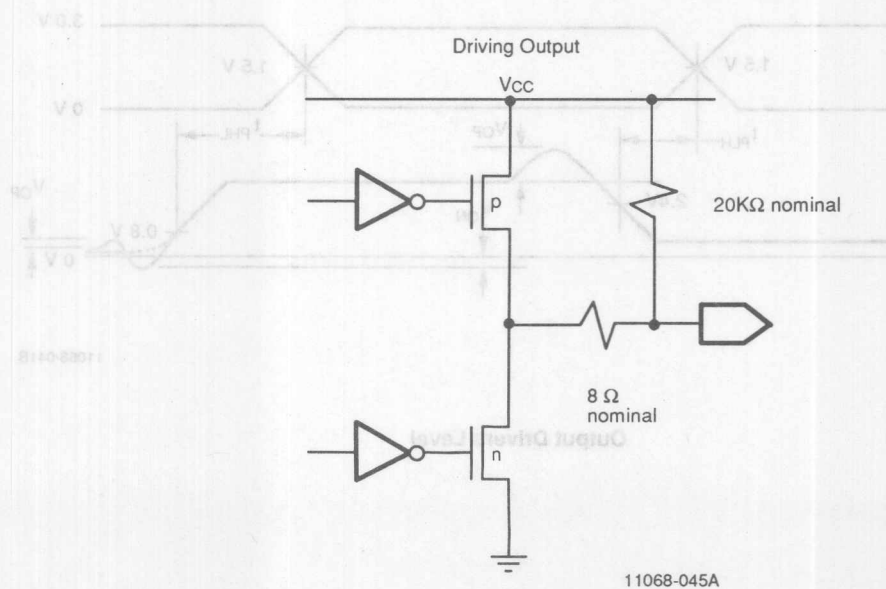
1. Ensure the part is adequately decoupled at the test head. Large changes in V_{CC} current as the device switches may cause erroneous function failures due to V_{CC} changes.
2. Do not leave inputs floating during any tests, as they may start to oscillate at high frequency.
3. Do not attempt to perform threshold tests at high speed. Following an input transition, ground current may change by as much as 400 mA in 5-8 ns. Inductance in the ground cable may allow the ground pin at the device to rise by hundreds of millivolts momentarily.
4. Use extreme care in defining input levels for AC tests. Many inputs may be changed at once, so there will be significant noise at the device pins and they may not actually reach V_{IL} or V_{IH} until the noise has settled. AMD recommends using $V_{IL} \leq 0$ V and $V_{IH} \geq 3$ V for AC tests.
5. To simplify failure analysis, programs should be designed to perform DC, Function, and AC tests as three distinct groups of tests.
6. Proper device grounding is critical when device testing. Multi-layer performance boards with radial decoupling between power and ground planes is recommended. Wiring unused interconnect pins to the ground plane is recommended. The ground plane must be sustained from the performance board to the device under test interface board. To minimize inductance, heavy-gauge stranded wire with twisted pairs should be used for power wiring.

SWITCHING WAVEFORMS

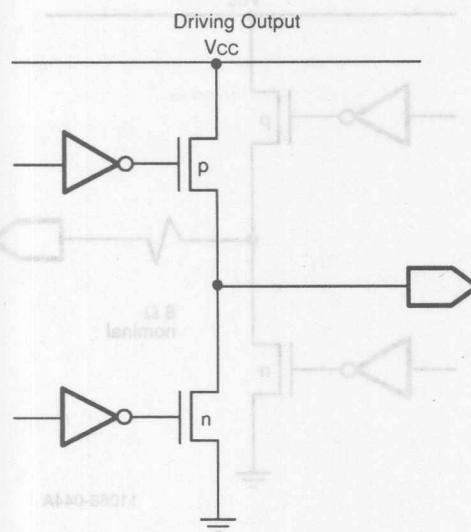




Equivalent Output Circuit
(Q_n Outputs)

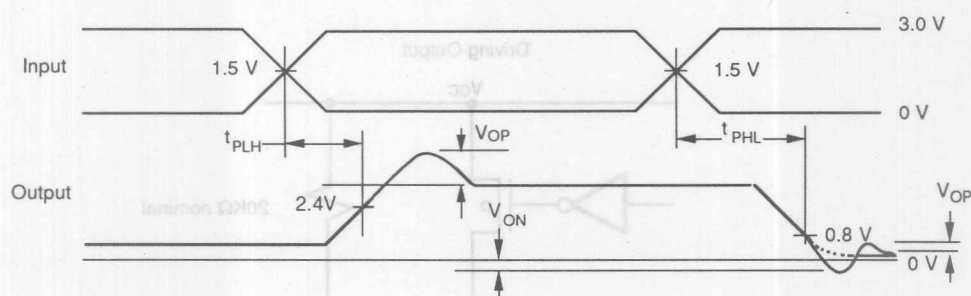


Equivalent Output Circuit
(RAS_n, CAS_n Outputs)



11068-046A

Equivalent Output Circuit
(All outputs except Q_n , \overline{RAS}_n , \overline{CAS}_n)



11068-041B

Output Drivers Level

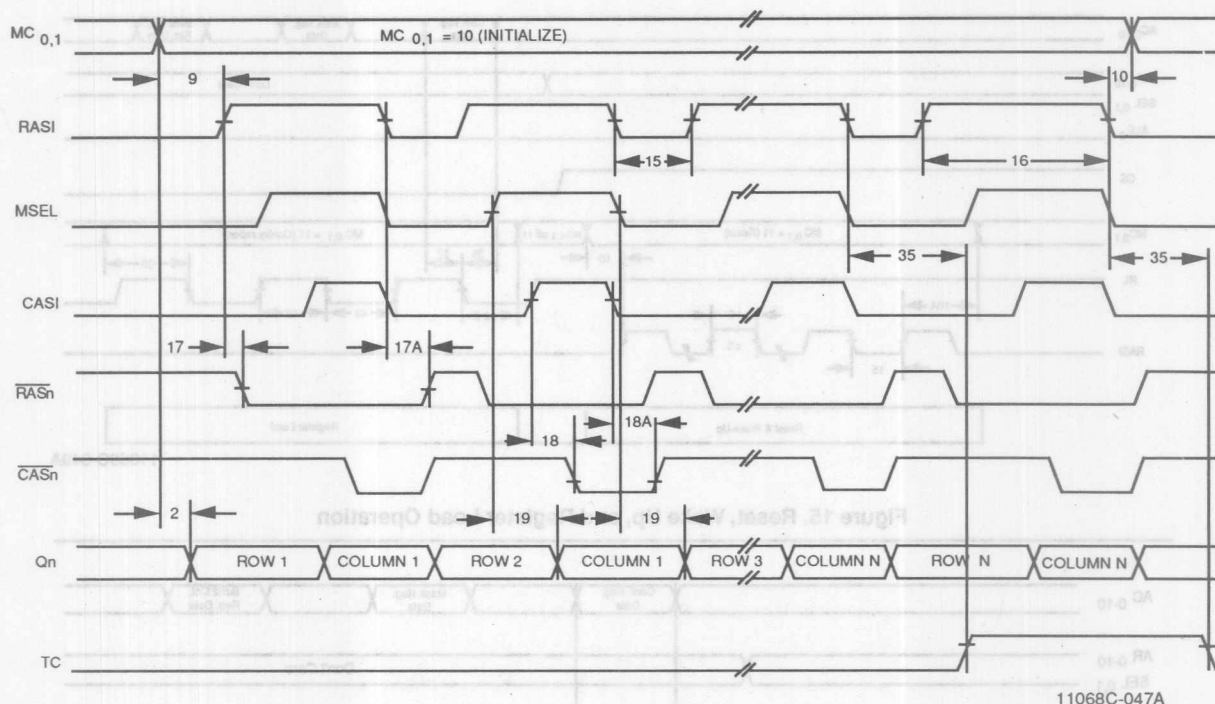


Figure 13. EDC Initialization with External Timing

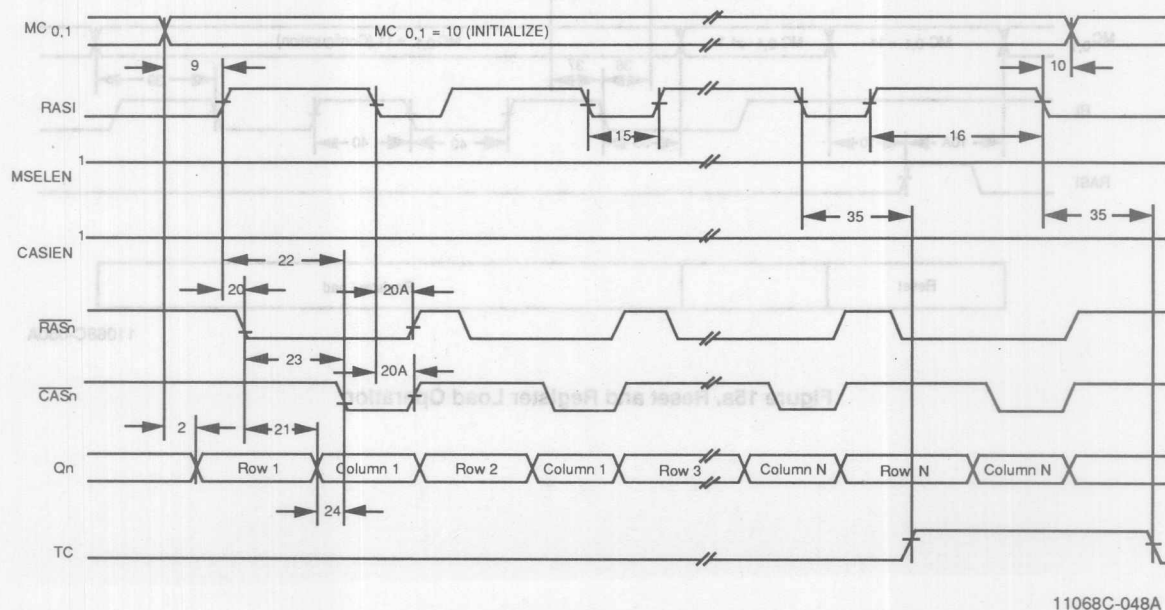


Figure 14. EDC Initialization with Auto-Timing

SWITCHING WAVEFORMS

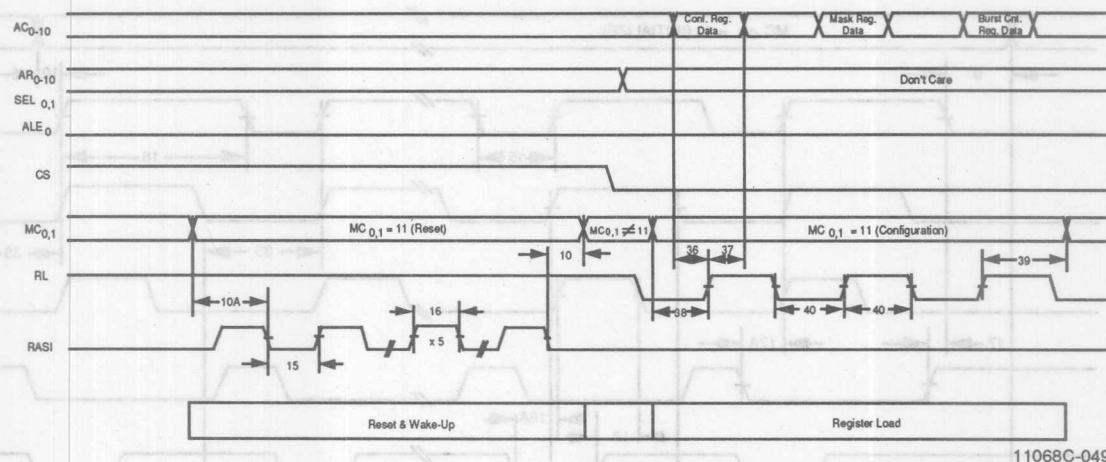


Figure 15. Reset, Wake Up, and Register Load Operation

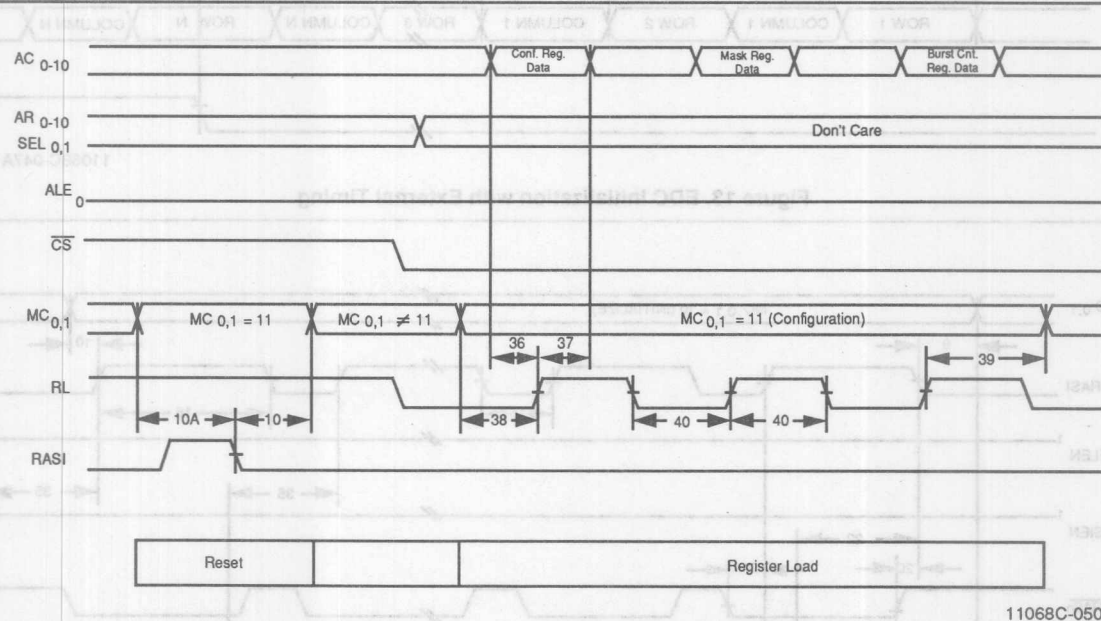
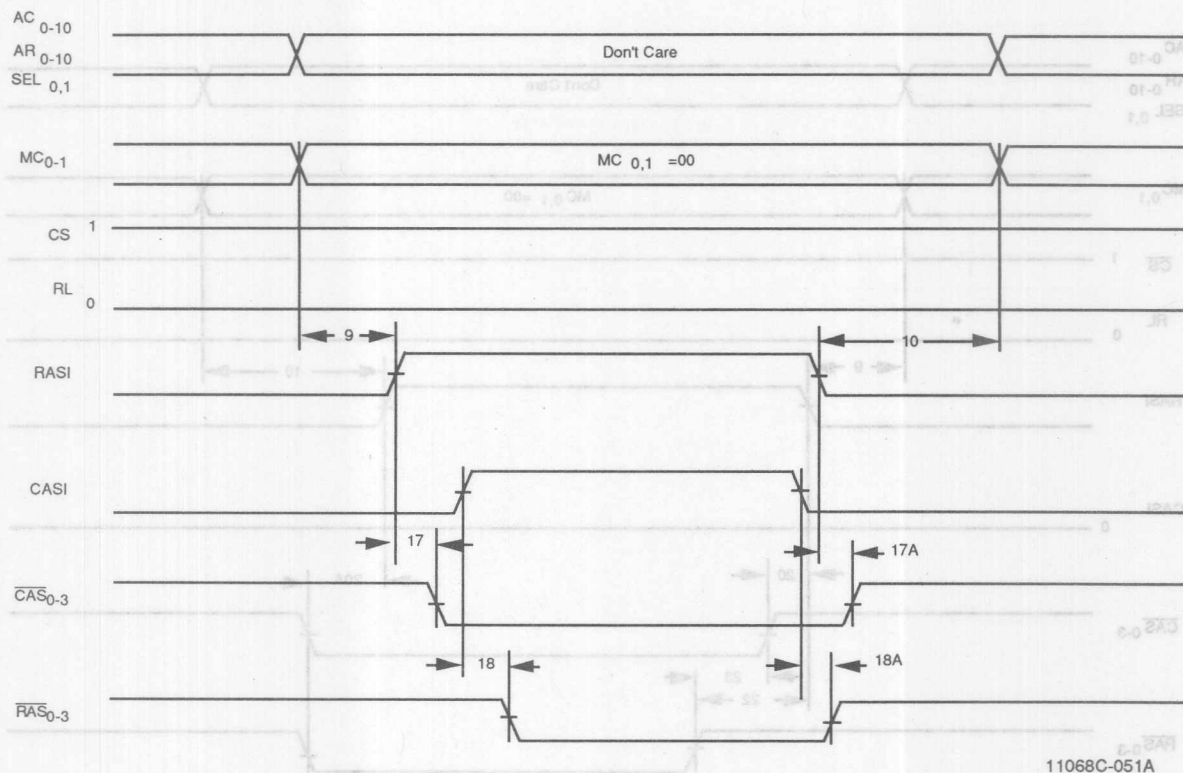


Figure 15a. Reset and Register Load Operation

SWITCHING WAVEFORMS



11068C-051A

Figure 15b. CAS Before RAS Refresh with External Timing

SWITCHING WAVEFORMS

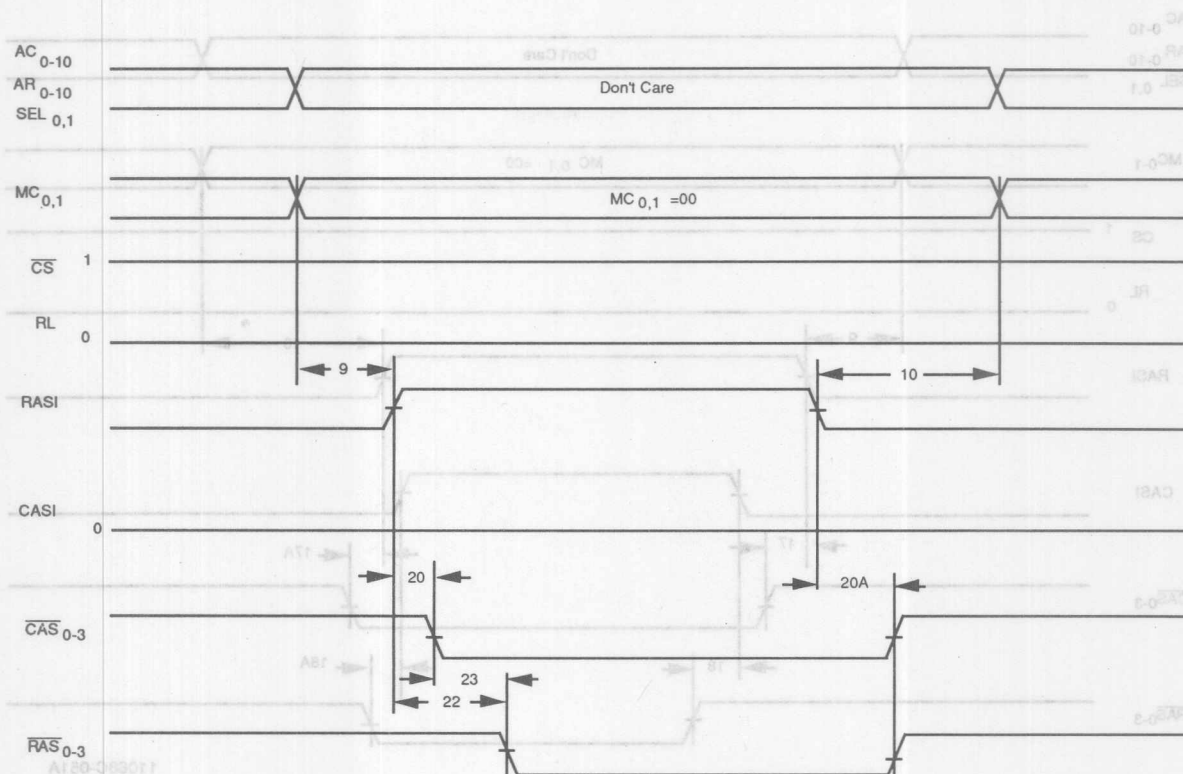
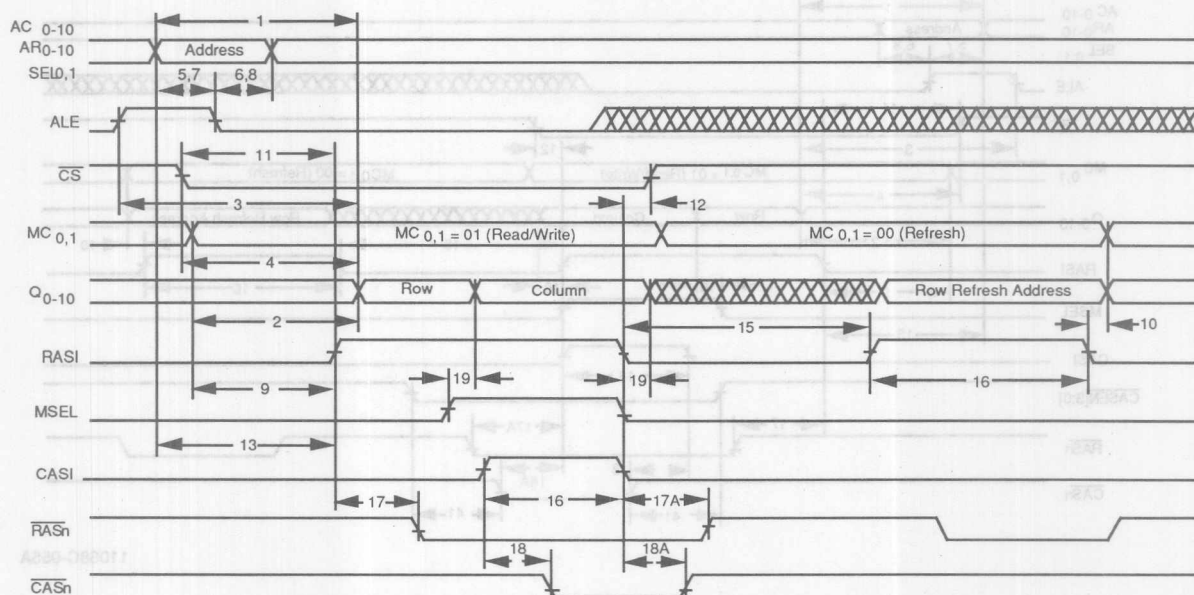


Figure 15c. CAS Before RAS Refresh with Auto Timing

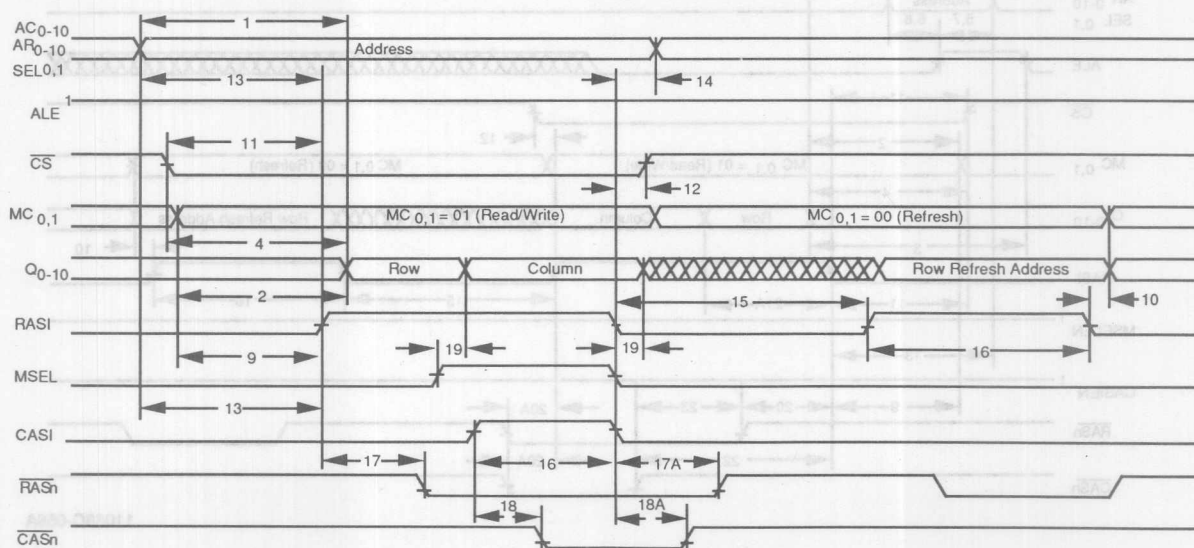
11068C-052A

SWITCHING WAVEFORMS



11068C-053A

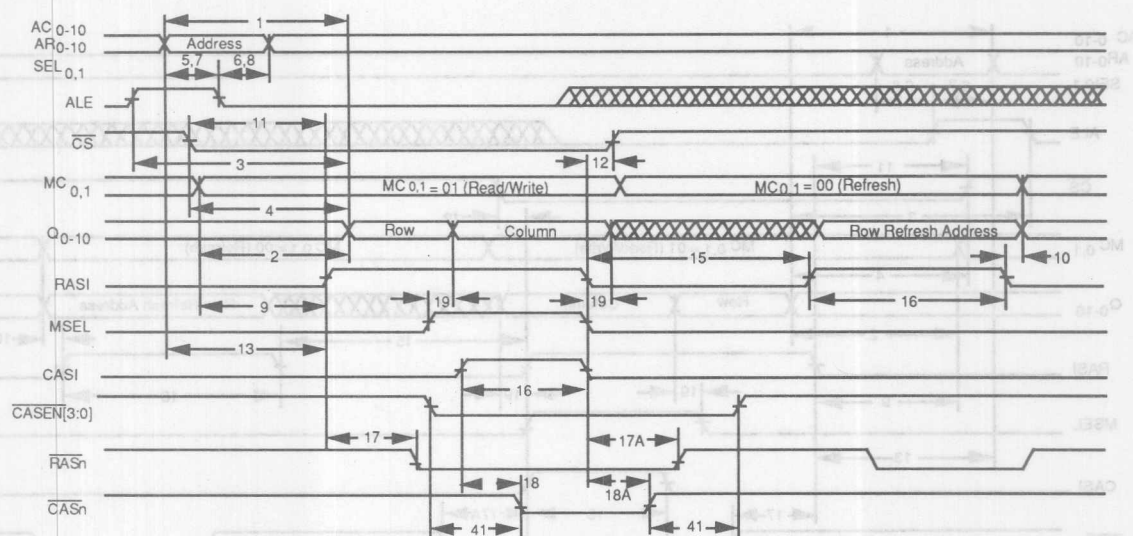
Figure 16. Standard Read/Write, Refresh Accesses with External Timing



11068C-054A

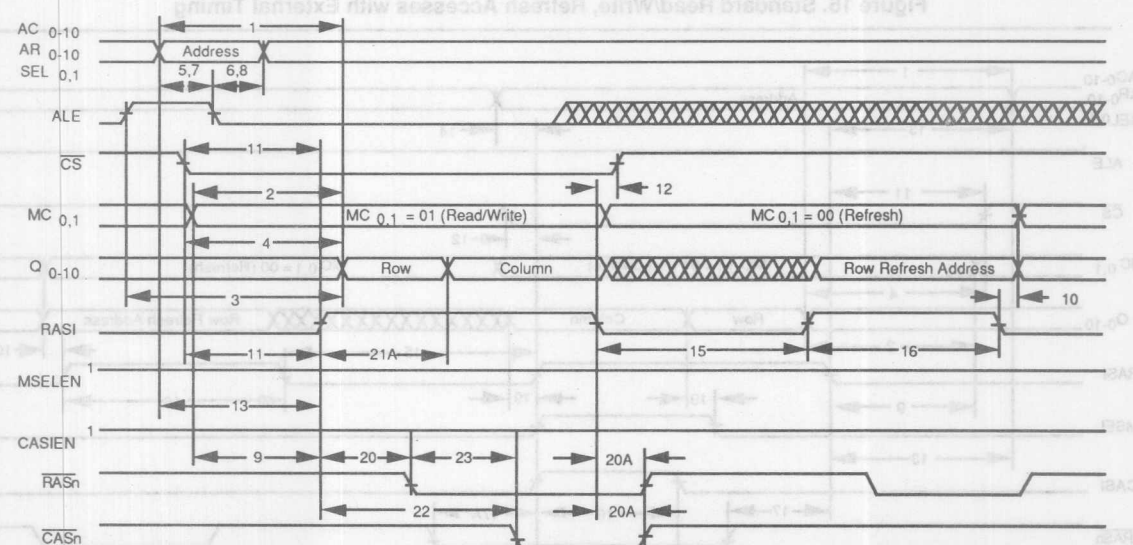
Figure 17. Standard Read/Write, Refresh Accesses with ALE = 1

SWITCHING WAVEFORMS



11068C-055A

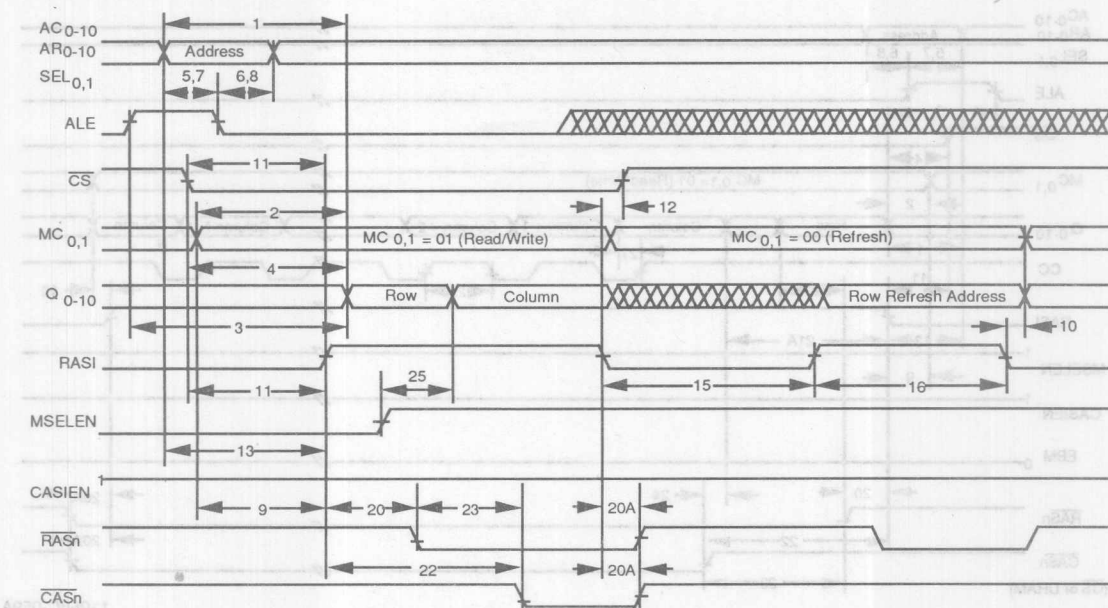
Figure 18. Standard Read/Write, Byte Access; Refresh (External Timing)



11068C-056A

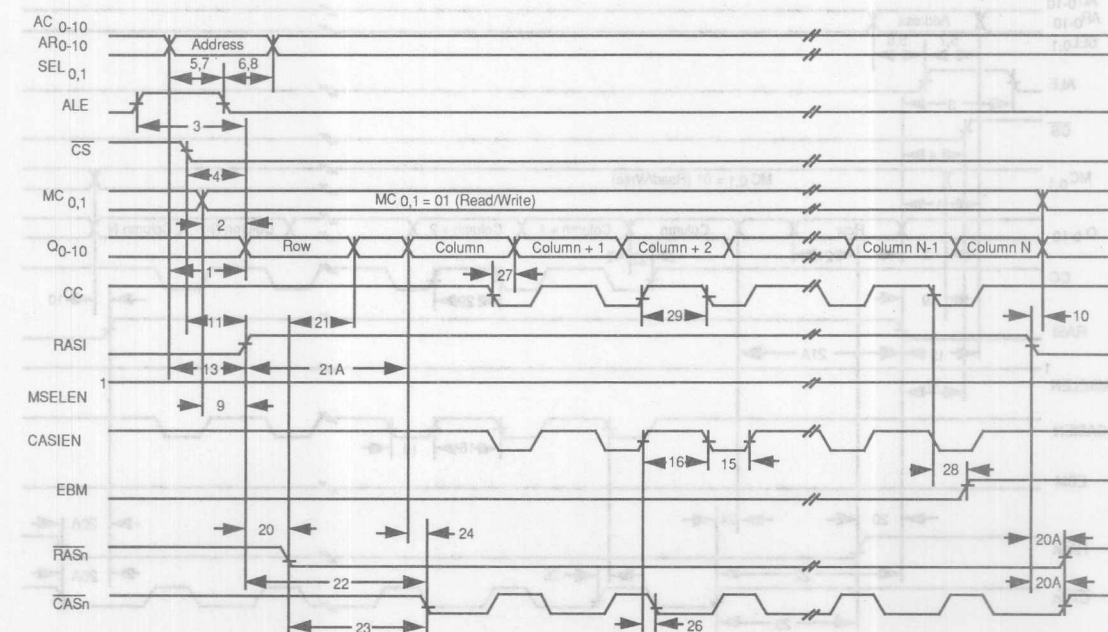
Figure 19. Standard Read/Write, Refresh Accesses with Auto-Timing

SWITCHING WAVEFORMS



11068C-057A

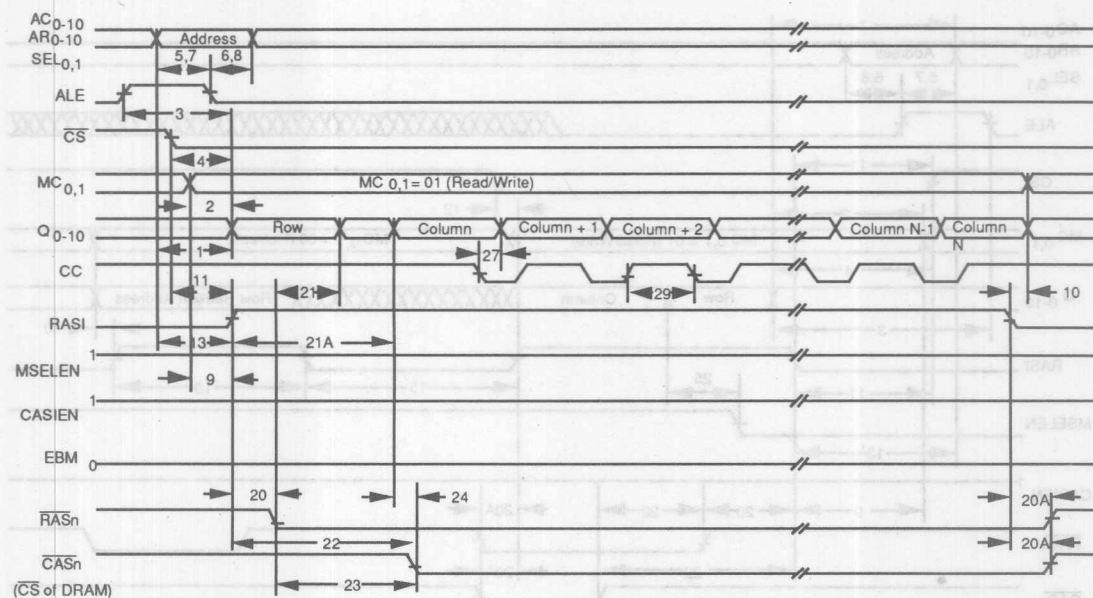
Figure 20. Read/Write, with Auto-Timing with External Override on MSELN



11068C-058A

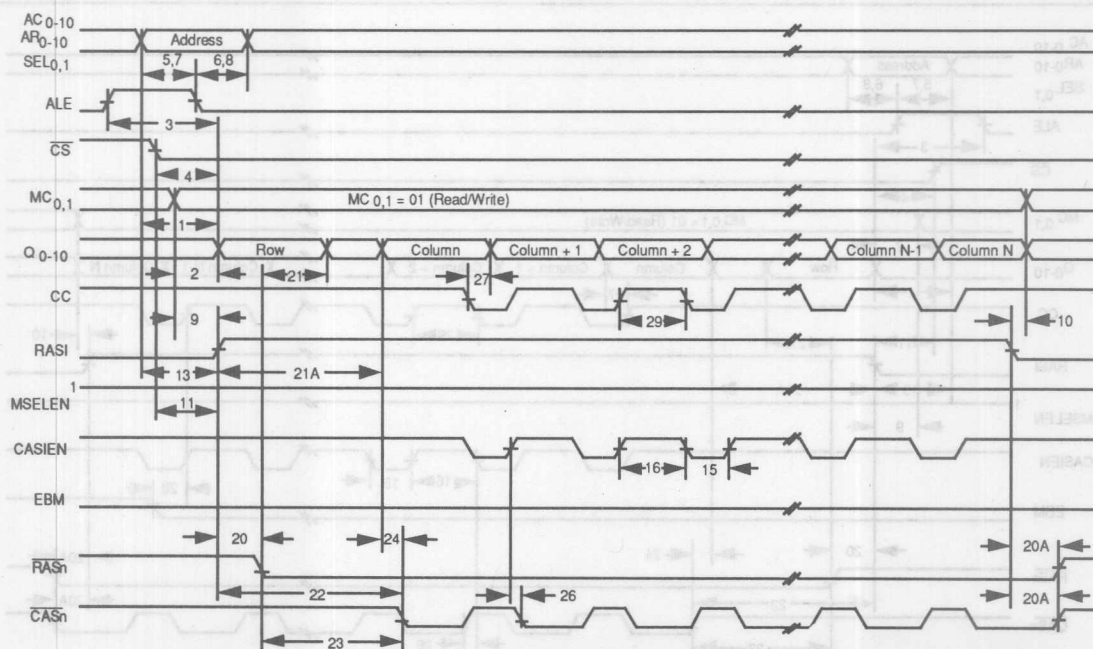
Figure 21. Burst Mode Access Ended by the Am29C668 (Auto-Timing with External Override on CAS_iEN)

SWITCHING WAVEFORMS



11068C-059A

Figure 22. Burst Mode Access with Static Column DRAMs Ended by the Microprocessor (Auto-Timing)



11068C-060A

Figure 23. Burst Mode Access Ended by the Microprocessor (Auto-Timing with External Override)

SWITCHING WAVEFORMS

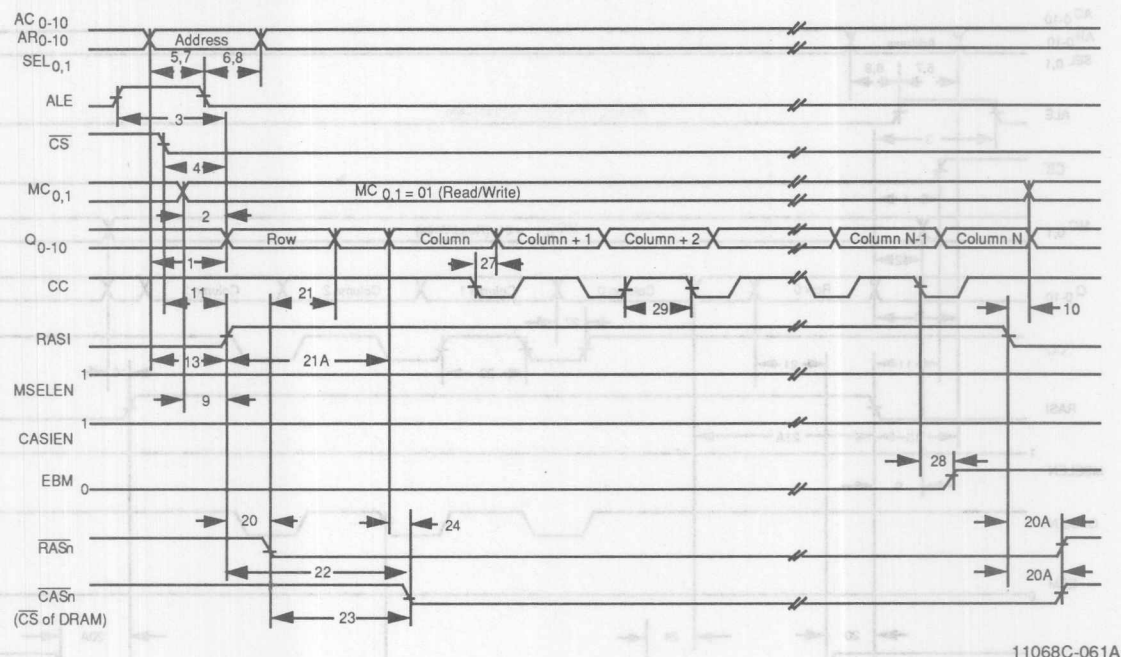


Figure 24. Burst Mode Access with Static Column DRAMs Ended by the Am29C668 (Auto-Timing)

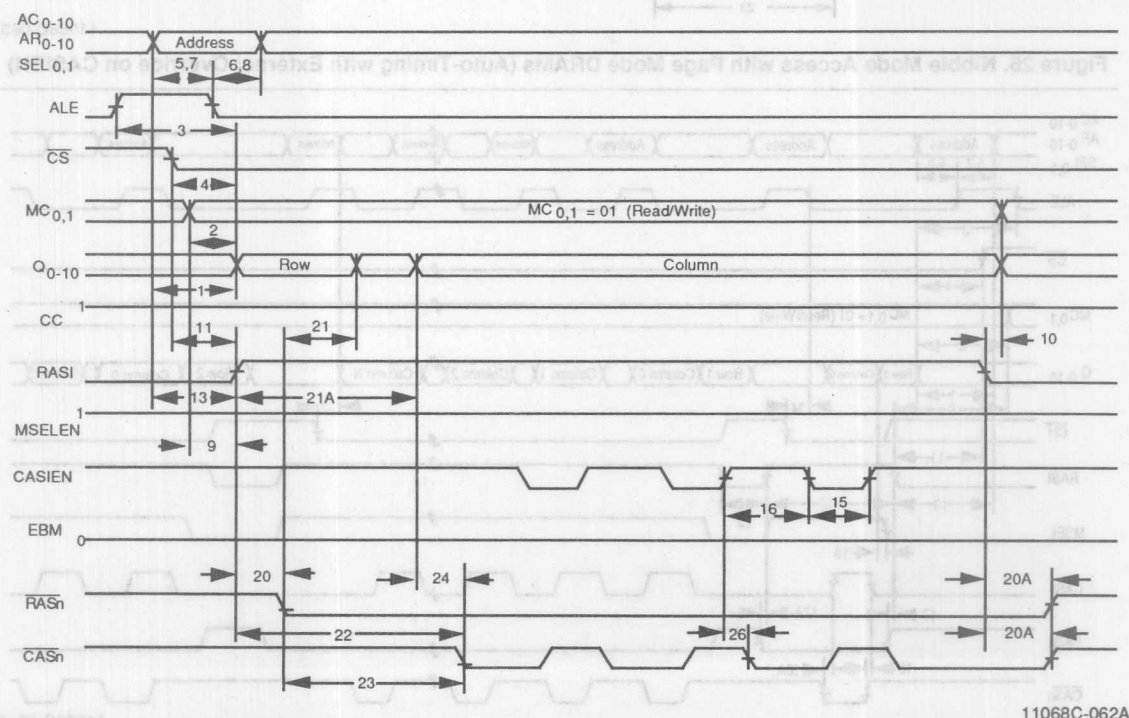
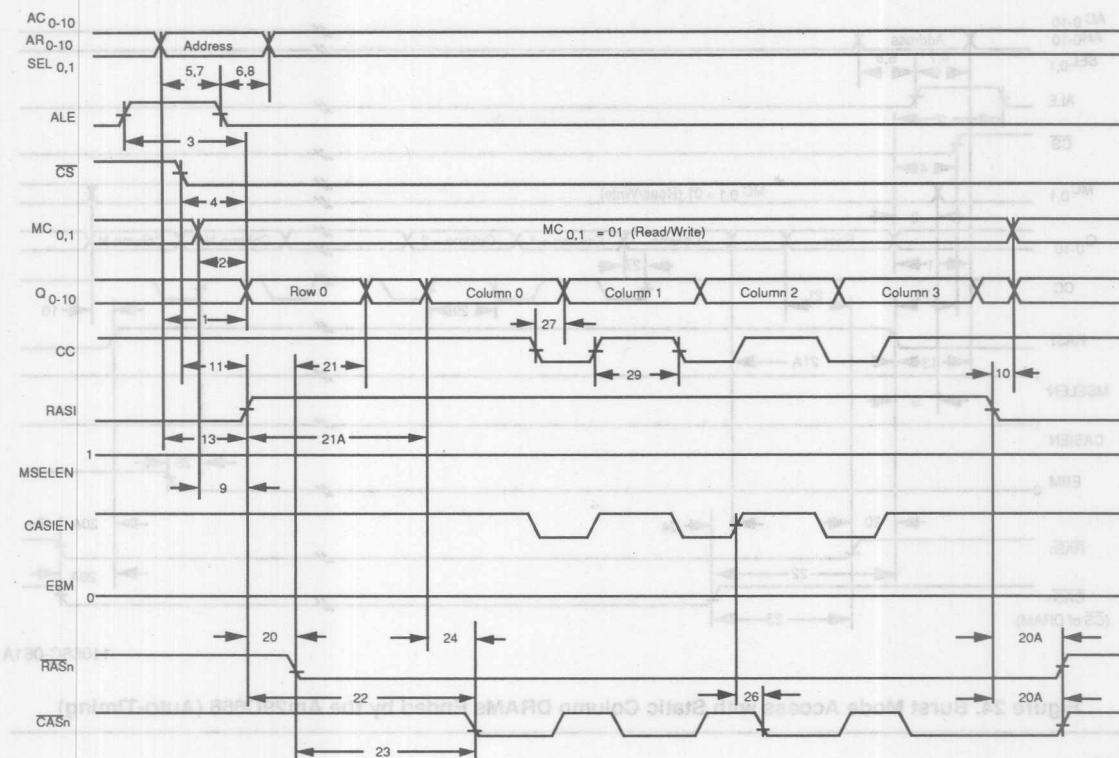


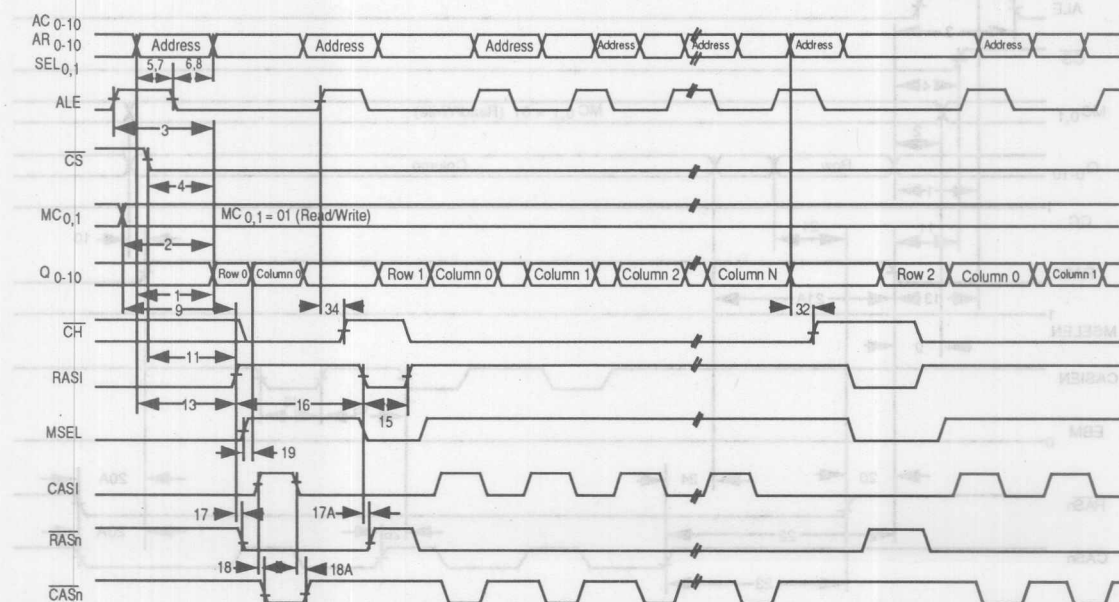
Figure 25. Burst Mode Access with Nibble Mode DRAMs (Auto-Timing with External Override)

SWITCHING WAVEFORMS



11068C-063A

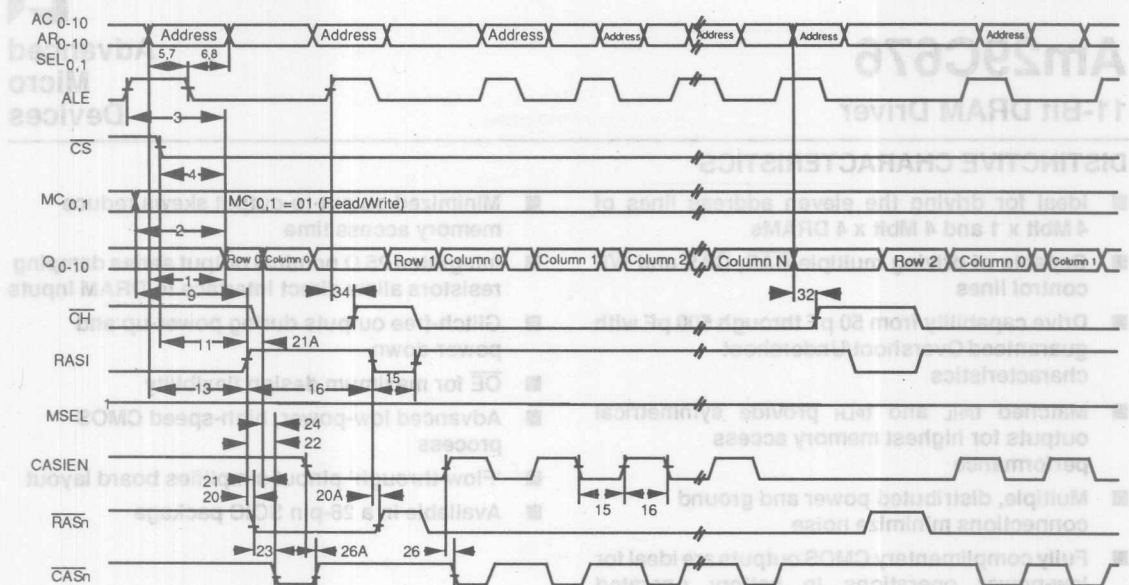
Figure 26. Nibble Mode Access with Page Mode DRAMs (Auto-Timing with External Override on CASIEN)



11068C-064A

Figure 27. "Cache" Mode Access with Page Mode DRAMs (External Timing)

SWITCHING WAVEFORMS



11068C-065A

Figure 28. "Cache" Mode Access with Page Mode DRAMs (Auto-Timing with External Override)

The Am29C668 incorporates power up/down circuitry which maintains the outputs in a high-impedance state during power supply sequencing. A voltage sensing circuit holds the outputs inactive (high impedance) when the voltage on V_{CC} is below a nominal value of 2.0 V. At V_{CC} values above this threshold, the outputs will respond to the steady-state input value. If they are enabled, in addition, each Am29C668 output structure employs a p-channel pull-up transistor that result in V_{OH} 2.7 V minimum to be compatible with MOS memory. The CMOS outputs of Am29C668 are fully complementary and draw minimum power during output transitions. This output structure is ideal for driving DRAMs in battery powered systems.

The Am29C668 is manufactured in an advanced CMOS logic process and is available in a 28-pin SOIC package.

The Am29C668 is a high-speed, high-drive 11-bit wide driver designed specifically to drive the address and control inputs of Dynamic RAMs. The non-inverting device is capable of driving all address lines for 256K through 4M x 1 DRAMs. The wide, 11-bit architecture minimizes skew times by consolidating all address paths through a single circuit, therefore, allowing reduced memory access times. In addition, up to 11 RAS and Write Enable control signals may be driven by a single Am29C668, thus reducing device count. An output enable allows for maximum design flexibility.

The Am29C668 uses 28Ω nominal internal series damping resistors to match the intrinsic trace impedance of memory systems. Proprietary output structures provide symmetrical HIGH-to-LOW and LOW-to-HIGH transitions, thus allowing for simplified, efficient design. The outputs are bounded by a specified maximum overshoot of +0.8 V and a maximum undershoot of -1.0 V from either a +5 V or 0 V reference.

Am29C676

11-Bit DRAM Driver

**Advanced
Micro
Devices**

DISTINCTIVE CHARACTERISTICS

- Ideal for driving the eleven address lines of 4 Mbit x 1 and 4 Mbit x 4 DRAMs
- Capable of driving multiple $\overline{\text{RAS}}$, $\overline{\text{CAS}}$ and $\overline{\text{WE}}$ control lines
- Drive capability from 50 pF through 500 pF with guaranteed Overshoot/Undershoot characteristics
- Matched tPHL and tPLH provide symmetrical outputs for highest memory access performance
- Multiple, distributed power and ground connections minimize noise
- Fully complimentary CMOS outputs are ideal for low-power operations in battery operated systems
- Minimized output-to-output skews reduce memory access time
- Integrated 25 Ω nominal output series damping resistors allow direct interface to DRAM inputs
- Glitch-free outputs during power-up and power-down
- $\overline{\text{OE}}$ for maximum design flexibility
- Advanced low-power, high-speed CMOS process
- 'Flow-through' pinout simplifies board layout
- Available in a 28-pin SOIC package

GENERAL DESCRIPTION

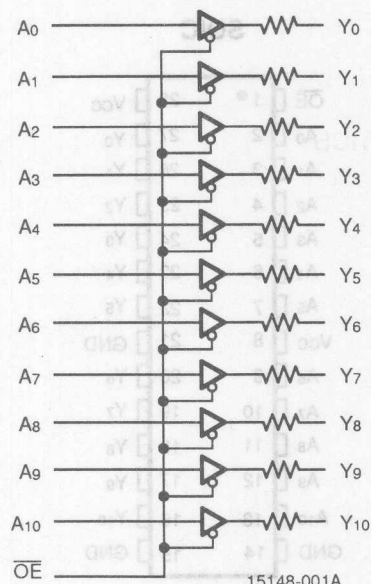
The Am29C676 is a high-speed, high-drive 11-bit wide driver designed specifically to drive the address and control inputs of Dynamic RAMs. The non-inverting device is capable of driving all address lines for 256K through 4M x n DRAMs. The wide, 11-bit architecture minimizes skew times by consolidating all address paths through a single circuit; therefore, allowing reduced memory access times. In addition, up to 11 $\overline{\text{RAS}}$, $\overline{\text{CAS}}$ and Write Enable control signals may be driven by a single Am29C676, thus reducing device count. An output enable allows for maximum design flexibility.

The Am29C676 uses 25 Ω nominal internal series damping resistors to match the intrinsic trace impedance of memory systems. Proprietary output structures provide symmetrical HIGH-to-LOW and LOW-to-HIGH transitions, thus allowing for simplified, efficient design. The outputs are bounded by a specified maximum overshoot of +0.8 V and a maximum undershoot of -1.0 V from either a +5 V or 0 V reference.

The Am29C676 incorporates power up/down circuitry which maintains the outputs in a high-impedance state during power supply sequencing. A voltage sensing circuit holds the outputs inactive (high impedance) when the voltage on V_{CC} is below a nominal value of 2.0 V. At V_{CC} values above this threshold, the outputs will respond to the steady-state input value, if they are enabled. In addition, each Am29C676 output structure employs p-channel pull-up transistors that result in $V_{OH} = 2.7$ V minimum to be compatible with MOS memory. The CMOS outputs of Am29C676 are fully complimentary and draw minimum power during output transitions. This output structure is ideal for driving DRAMs in battery powered systems.

The Am29C676 is manufactured in an advanced CMOS logic process and is available in a 28-pin SOIC package.

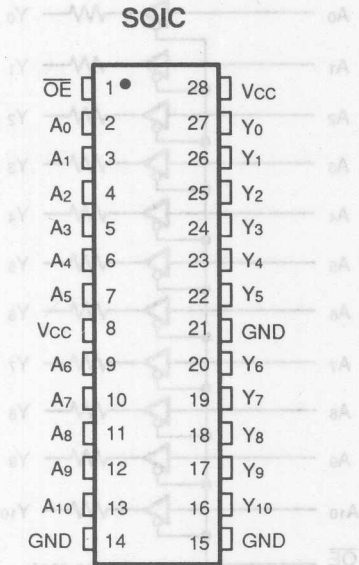
BLOCK DIAGRAM



RELATED AMD PRODUCTS

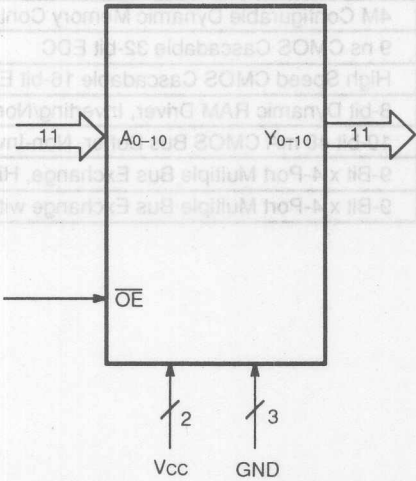
Part No.	Description
Am29C668	4M Configurable Dynamic Memory Controller/Driver
Am29C660E	9 ns CMOS Cascadable 32-bit EDC
Am29C60A	High Speed CMOS Cascadable 16-bit EDC
Am2965/66	8-bit Dynamic RAM Driver, Inverting/Non-Inverting
Am29C827A/28A	10-bit 48 mA CMOS Bus Buffer, Non-Inverting/Inverting
Am29C983A	9-Bit x 4-Port Multiple Bus Exchange, High Speed
Am29C985	9-Bit x 4-Port Multiple Bus Exchange with Parity

CONNECTION DIAGRAM
Top View



15148-002A

LOGIC SYMBOL



Die Size: 0.079" x 0.138"
Gate Count: 66

15148-004A

PACKAGE INFORMATION

Parameter	SOIC	Unit
$\varnothing JA$	55	$^{\circ}C/W$

FUNCTION TABLE

Inputs		Outputs	Function
\overline{OE}	A_i	Y_i	
L	H	H	Transparent
L	L	L	Transparent
H	X	Z	Hi-Z

ORDERING INFORMATION

Standard Products

AMD standard products are available in several packages and operating ranges. The order number (Valid Combination) is formed by a combination of these elements:

Device Number
Speed Option (If applicable)
Package Type
Temperature Range
Optional Processing

Function	Input	Output	OE
Transparent	H	H	L
Transparent	L	L	L
Transparent	X	X	H

AM29C676

S

C

OPTIONAL PROCESSING

Blank = Standard processing

TEMPERATURE RANGE

C = Commercial (0 to +70°C)

PACKAGE TYPE

S = 28-Pin (300-mil) Plastic Small Outline
Package (SO 028)

SPEED OPTION

Not Applicable

DEVICE NUMBER/DESCRIPTION

Am29C676
11-Bit DRAM Driver

Valid Combinations

AM29C676	SC
----------	----

Valid Combinations

Valid Combinations list configurations planned to be supported in volume for this device. Consult the local AMD sales office to confirm availability of specific valid combinations or to check on newly released combinations, and to obtain additional data on AMD's standard military grade products.

PIN DESCRIPTION

A₀ – A₁₀**Driver Address Inputs (Inputs (11))**

These eleven pins are the inputs of the driver.

Y₀ – Y₁₀**Driver Address Outputs (Outputs (11))**

These edge-rate controlled non-inverting outputs directly drive the DRAM address inputs. The drivers on these lines are capable of driving high capacitive loads, from 50 pF to over 500 pF.

 $\overline{\text{OE}}$ **Output Enable (Input; Active LOW)**

When in the LOW state, the three-state output lines are enabled. When this input is HIGH, the outputs are in the high-impedance state.

GND**0 V Power Supply (3)**

All GND pins must be connected for proper device operation.

V_{cc}**+5 V Power Supply (2)**

All V_{cc} pins must be connected for proper device operation.

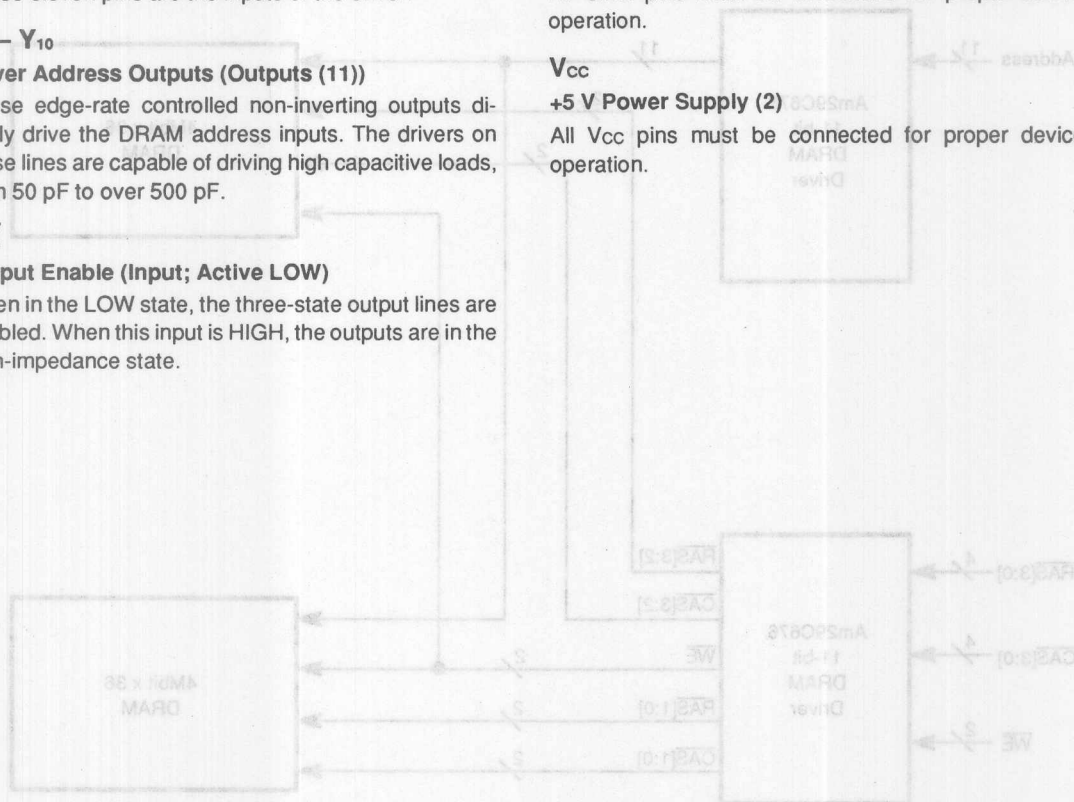
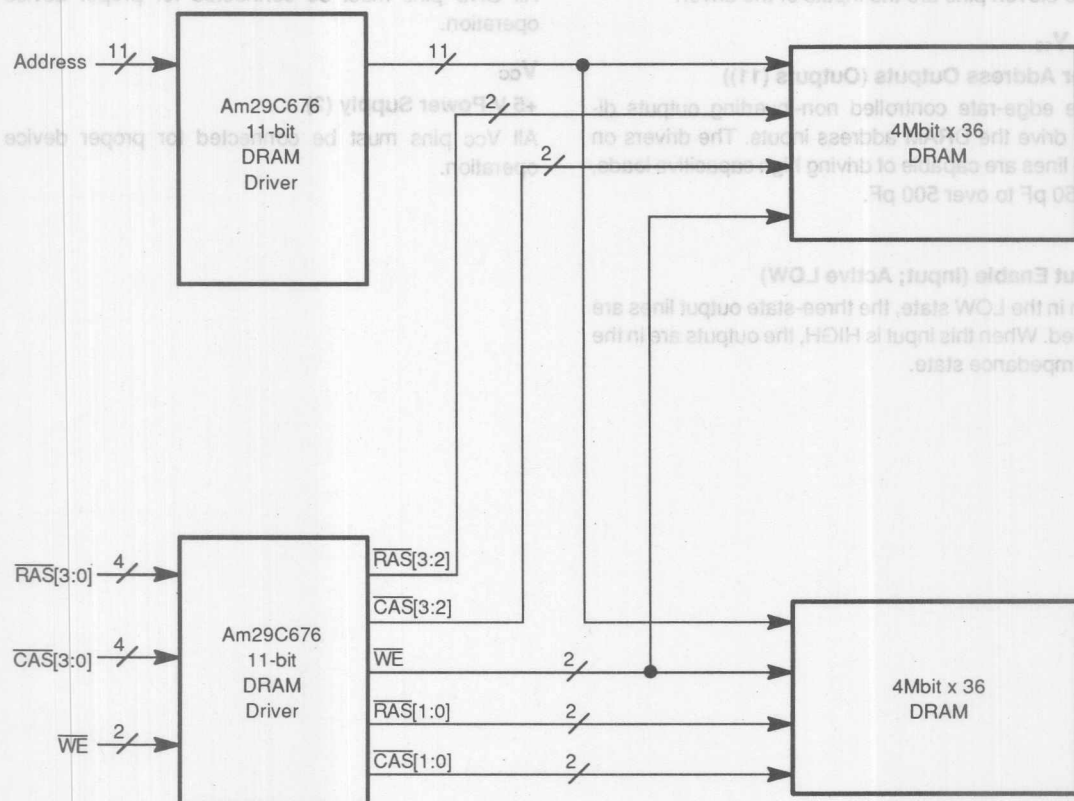


Figure 1: Am29C676 Two-Bank System Application
Only two drivers are needed to address and control 32 Mbytes of DRAM. Pairs of FAS and WE lines are tied together to reduce loading and decrease propagation delay (reference Figure 4).

APPLICATIONS

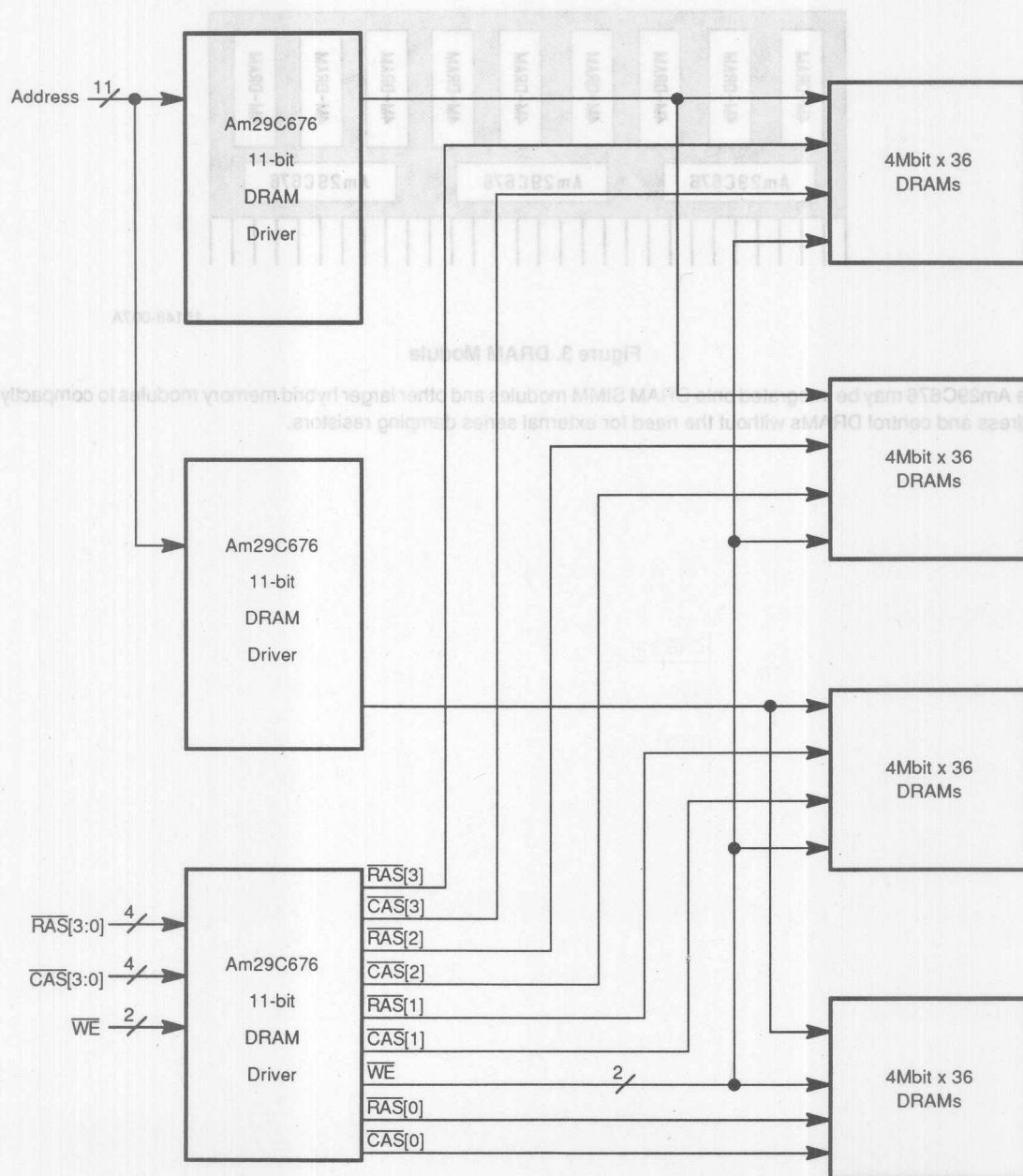


15148-005A

Figure 1. Am29C676 Two-Bank System Application

Only two drivers are needed to address and control 32 Mbytes of DRAM. Pairs of $\overline{\text{RAS}}$ and $\overline{\text{WE}}$ lines are tied together to reduce loading and decrease propagation delay (reference Figure 4).

APPLICATIONS



15148-006A

Figure 2. Am29C676 Four-Bank System Application

Only three drivers are needed to address and control 64 Mbytes of DRAM

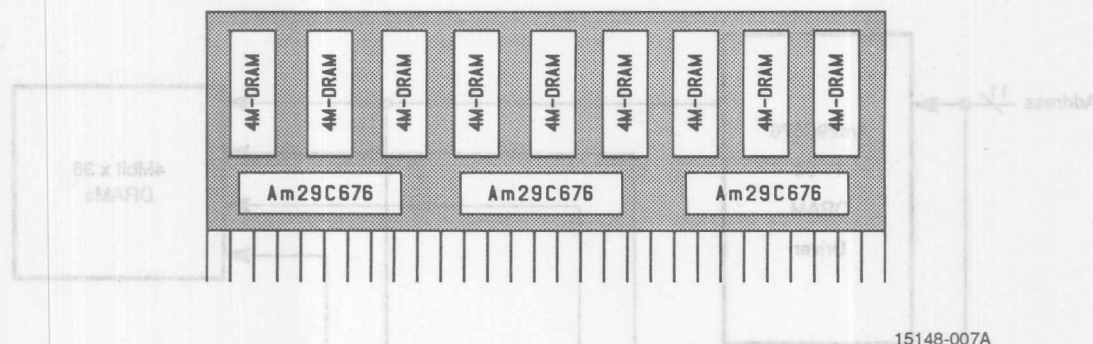


Figure 3. DRAM Module

The Am29C676 may be integrated onto DRAM SIMM modules and other larger hybrid memory modules to compactly address and control DRAMs without the need for external series damping resistors.

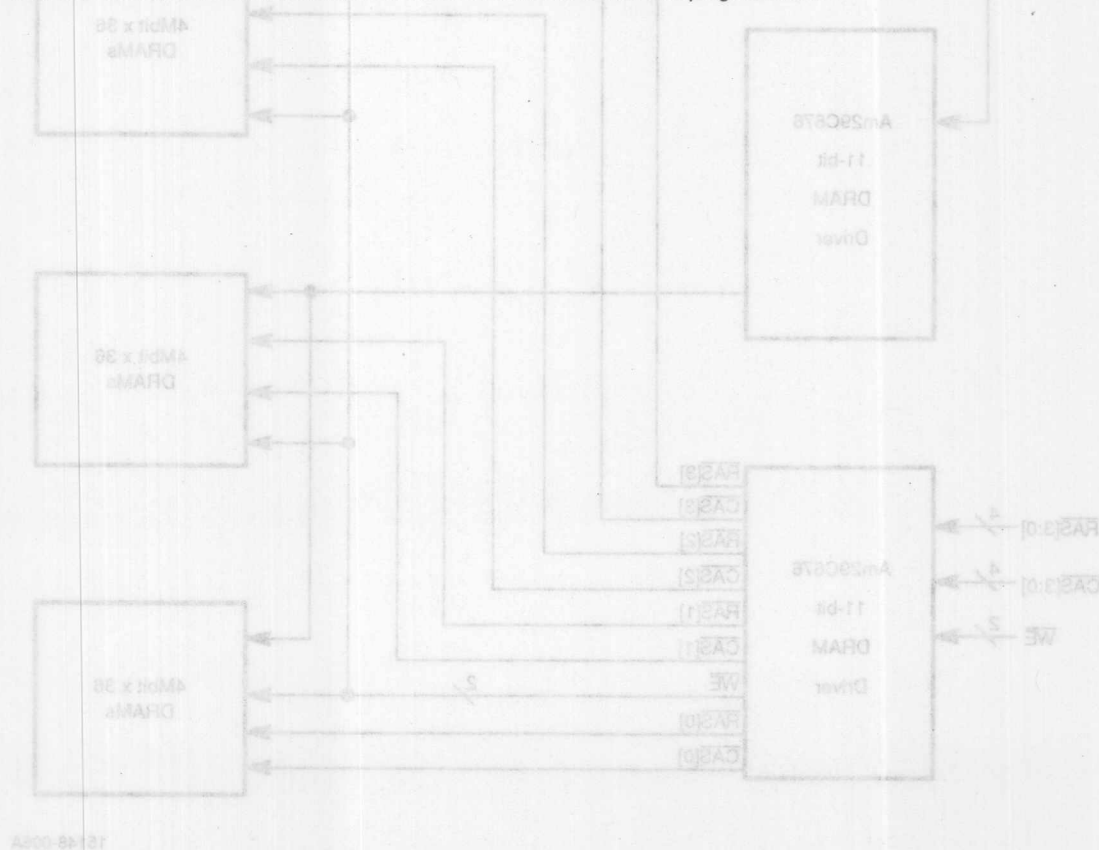


Figure 5. Am29C676 Four-Bank System Application

Only three drivers are needed to address and control 64 Mbytes of DRAM.

ABSOLUTE MAXIMUM RATINGS

Storage Temperature	-65 to 150°C
Supply Voltage to Ground	
Potential Continuous	-0.5 V to +6.0 V
DC Output Voltage	-0.5 V to (V _{CC} + 0.3) V
DC Input Voltage	-0.5 V to (V _{CC} + 0.3) V
DC Output Diode Current	-50 mA
DC Input Diode Current	-20 mA
DC Output Current per Pin:	

I _{SINK}	+70 mA
I _{SOURCE}	-30 mA

Total DC GND Current	
(n x I _{OL} + m x I _{CC}) mA	(Note 1)
Total DC V _{CC} Current	
(n x I _{OH} + m x I _{CC}) mA	(Note 1)

Note:

1. n = number of outputs, m = number of inputs.

Stresses above those listed under ABSOLUTE MAXIMUM RATINGS may cause permanent device failure. Functionality at or above these limits is not implied. Exposure to absolute maximum ratings for extended periods may affect device reliability.

OPERATING RANGES

Commercial (C) Devices

Temperature (T _A)	0 to +70°C
Supply Voltage (V _{CC})	+4.5 V to +5.5 V

Operating ranges define those limits between which the functionality of the device is guaranteed.

V _{CC}	4.5 V to 5.5 V	Output LOW Voltage	V _{OL}
V _{CC}	4.5 V to 5.5 V	Output HIGH Voltage	V _{OH}
V _{CC}	4.5 V to 5.5 V	Input LOW Level	V _{IL}
V _{CC}	4.5 V to 5.5 V	Input HIGH Level	V _{IH}
V _{CC}	4.5 V to 5.5 V	Input Clamp Voltage	V _I
V _{CC}	4.5 V to 5.5 V	Output Underfoot	V _{ON}
V _{CC}	4.5 V to 5.5 V	Output Overfoot	V _{OP}
V _{CC}	4.5 V to 5.5 V	Input LOW Current	I _{IL}
V _{CC}	4.5 V to 5.5 V	Input HIGH Current	I _{IH}
V _{CC}	4.5 V to 5.5 V	Off-State Output Current (I _{OS})	I _{OS}
V _{CC}	4.5 V to 5.5 V	Output Short-Circuit Current	I _{SC}
V _{CC}	4.5 V to 5.5 V	Static Supply Current	I _{CC}
V _{CC}	4.5 V to 5.5 V	Dynamic Supply Current	I _{CCD}

Notes:

1. Input thresholds are tested in combination with other DC parameters or by correlation.
2. Not more than one output should be strobed at a time. Duration should not exceed 100 milliseconds.
3. Measured at frequency ≤ 10 MHz with 50% duty cycle.
4. V_{ON} and V_{OP} are not production tested but are guaranteed by characterization for surface mounted devices with proper capacitive decoupling. Limits specified are for all outputs switching simultaneously with minimum specified loading. As loading increases, V_{ON} and V_{OP} will approach zero. Reference Switching Waveforms.

CAPACITANCE (Note 1)

Parameter Symbol	Parameter Description	Test Conditions	Typ.	Unit
C _{IN}	Input Capacitance	f = 1 MHz T _A = 25°C	8	pF
C _{OUT}	Output Capacitance		10	pF

Note:

1. These parameters are not tested in production and are not guaranteed, but are evaluated at initial characterization and at any time the design is modified where capacitance may be affected.

DC CHARACTERISTICS over operating range unless otherwise specified**V_{CC} = 5 V ± 10%**

Parameter Symbol	Parameter Description	Test Conditions			Min.	Max.	Unit
V _{OH}	Output HIGH Voltage	V _{CC} = 4.5 V	I _{OH} = −10 mA		2.7		V
		V _{IN} = V _{IH} or V _{IL}	I _{OH} = −500 μA		V _{CC} − 0.2		V
V _{OL}	Output LOW Voltage	V _{CC} = 4.5 V	I _{OL} = 10 mA			0.5	V
		V _{IN} = V _{IH} or V _{IL}	I _{OL} = 500 μA			0.2	V
V _{IH}	Input HIGH Level	Guaranteed input logical HIGH voltage for all inputs (Note 1)			2.0		V
V _{IL}	Input LOW Level	Guaranteed input logical LOW voltage for all inputs (Note 1)				0.8	V
V _I	Input Clamp Voltage	V _{CC} = 4.5 V, I _{IN} = −18 mA				−1.2	V
V _{ON}	Output Undershoot H-to-L Voltage (Note 4)	C _L = 50 pF				−1.0	V
V _{OP}	Output Overshoot H-to-L Voltage (Note 4)	C _L = 50 pF				0.8	V
V _{OP}	Output Overshoot L-to-H Voltage (Note 4)	C _L = 50 pF				0.8	V
I _{IL}	Input LOW Current	V _{CC} = 5.5 V, V _{IN} = GND				−10	μA
I _{IH}	Input HIGH Current	V _{CC} = 5.5 V, V _{IN} = 5.5 V				10	μA
I _{OZ}	Off-State Output Current (Hi-Z)	V _{CC} = 5.5 V, V _O = 5.5 V				+20	μA
		V _{CC} = 5.5 V, V _O = GND				−20	μA
I _{SC}	Output Short-Circuit Current	V _{CC} = 5.5 V, V _O = 0 V (Note 2)			−60	−180	mA
I _{CCQ}	Static Supply Current	V _{CC} = 5.5 V Outputs Open	V _{IN} = V _{CC} or GND			1.2	mA
I _{CCT}			V _{IN} = 2.4 V	Data Input		4.0	mA/Bit
				$\overline{\text{OE}}$		4.5	mA
I _{CCD}	Dynamic Supply Current	V _{CC} = 5.5 V (Note 3)	Outputs Open			300	μA/MHz
			Outputs Loaded	C _L = 50 pF		440	/Bit

Notes:

- Input thresholds are tested in combination with other DC parameters or by correlation.
- Not more than one output should be shorted at a time. Duration should not exceed 100 milliseconds.
- Measured at a frequency ≤ 10 MHz with 50% duty cycle.
- V_{ON} and V_{OP} are not production tested but are guaranteed by characterization for surface mounted devices with proper capacitive decoupling. Limits specified are for all outputs switching simultaneously with minimum specified loading. As loading increases, V_{ON} and V_{OP} will approach zero. Reference Switching Waveforms.

CAPACITANCE (Note 1)

Parameter Symbol	Parameter Description	Test Conditions	Typ.	Unit
C _{IN}	Input Capacitance	f = 1 MHz	6	pF
C _{OUT}	Output Capacitance	T _A = 25°C	10	pF

Note:

- These parameters are not tested in production and are not guaranteed, but are evaluated at initial characterization and at any time the design is modified where capacitance may be effected.

DC CHARACTERISTICS over COMMERCIAL operating range unless otherwise specified
 $V_{CC} = 3.3 \text{ V} \pm 10\%$

ADVANCED INFORMATION						
Parameter Symbol	Parameter Description	Test Conditions		Min.	Max.	Unit
V _{OH}	Output HIGH Voltage	V _{CC} = 3.0 V	I _{OH} = −2 mA	2.4		V
		V _{IN} = V _{IH} or V _{IL}	I _{OH} = −20 μA	V _{CC} − 0.1		V
V _{OL}	Output LOW Voltage	V _{CC} = 3.0 V	I _{OL} = 4 mA		0.4	V
		V _{IN} = V _{IH} or V _{IL}	I _{OL} = 20 μA		0.1	V
V _{IH}	Input HIGH Level	Guaranteed input logical HIGH voltage for all inputs (Note 1)		2.0		V
V _{IL}	Input LOW Level	Guaranteed input logical LOW voltage for all inputs (Note 1)			0.8	V
V _I	Input Clamp Voltage	V _{CC} = 3.0 V, I _{IN} = −18 mA			−1.2	V
V _{ON}	Output Undershoot H-to-L Voltage (Note 4)	C _L = 50 pF			−1.0	V
V _{OP}	Output Overshoot H-to-L Voltage (Note 4)	C _L = 50 pF			0.8	V
V _{OP}	Output Overshoot L-to-H Voltage (Note 4)	C _L = 50 pF			0.8	V
I _{IL}	Input LOW Current	V _{CC} = 3.6 V, V _{IN} = GND			−10	μA
I _{IH}	Input HIGH Current	V _{CC} = 3.6 V, V _{IN} = 3.6 V			10	μA
I _{OZ}	Off-State Output	V _{CC} = 3.6 V, V _O = 3.6 V			+20	μA
	Current (Hi-Z)	V _{CC} = 3.6 V, V _O = GND			−20	μA
I _{SC}	Output Short-Circuit Current	V _{CC} = 3.6 V, V _O = 0 V (Note 2)		—	—	mA
I _{CCQ}	Static Supply Current	V _{CC} = 3.6 V Outputs Open	V _{IN} = V _{CC} or GND		—	mA
I _{CCT}			V _{IN} = 2.4 V	Data Input	—	mA/Bit
				OE	—	mA
I _{CCD}	Dynamic Supply Current	V _{CC} = 3.6 V (Note 3)	Outputs Open		—	μA/MHz
			Outputs Loaded	C _L = 50 pF	—	/Bit

Notes:

1. Input thresholds are tested in combination with other DC parameters or by correlation.
2. Not more than one output should be shorted at a time. Duration should not exceed 100 milliseconds.
3. Measured at a frequency $\leq 10 \text{ MHz}$ with 50% duty cycle.
4. V_{ON} and V_{OP} are not production tested but are guaranteed by characterization for surface mounted devices with proper capacitive decoupling. Limits specified are for all outputs switching simultaneously with minimum specified loading. As loading increases, V_{ON} and V_{OP} will approach zero. Reference Switching Waveforms.

SWITCHING CHARACTERISTICS over COMMERCIAL operating range unless otherwise specified **$V_{CC} = 5\text{ V} \pm 10\%$**

No.	Parameter Symbol	Parameter Description	Test Conditions*	Min.	Max.	Unit
1	t _{PLH}	Address (A _i) to Output (Y _i)	C _L = 50 pF		7	ns
2	t _{PHL}				7	ns
3	t _{ZH}	Output Enable Time			11	ns
4	t _{ZL}	$\overline{\text{OE}}$ to Y _i			11	ns
5	t _{HZ}	Output Disable Time			10	ns
6	t _{LZ}	$\overline{\text{OE}}$ to Y _i			10	ns

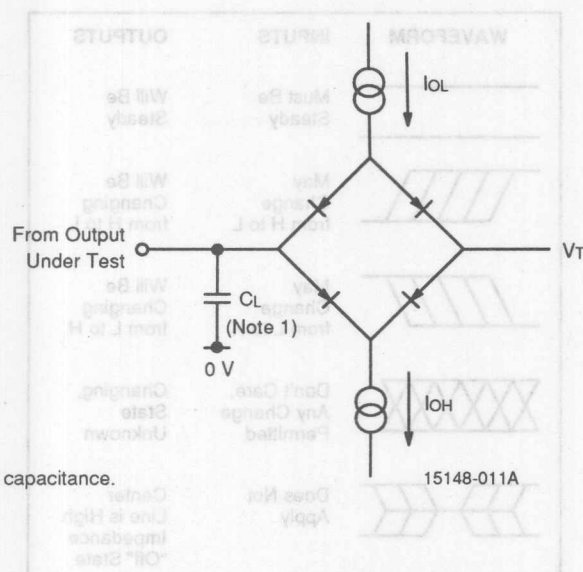
 $V_{CC} = 3.3\text{ V} \pm 10\%$

ADVANCED INFORMATION						
No.	Parameter Symbol	Parameter Description	Test Conditions	Min.	Max.	Unit
1	t _{PLH}	Address (A _i) to Output (Y _i)	C _L = 50 pF		—	ns
2	t _{PHL}				—	ns
3	t _{ZH}	Output Enable Time			—	ns
4	t _{ZL}	$\overline{\text{OE}}$ to Y _i			—	ns
5	t _{HZ}	Output Disable Time			—	ns
6	t _{LZ}	$\overline{\text{OE}}$ to Y _i			—	ns

*See Test Circuit and Waveforms. For performance at higher loads, reference Figure 4.

ADVANCED INFORMATION						
No.	Parameter Symbol	Parameter Description	Test Conditions	Min.	Max.	Unit
1	t _{PLH}	Address (A _i) to Output (Y _i)	C _L = 50 pF		—	ns
2	t _{PHL}				—	ns
3	t _{ZH}	Output Enable Time			—	ns
4	t _{ZL}	$\overline{\text{OE}}$ to Y _i			—	ns
5	t _{HZ}	Output Disable Time			—	ns
6	t _{LZ}	$\overline{\text{OE}}$ to Y _i			—	ns

SWITCHING TEST CIRCUIT

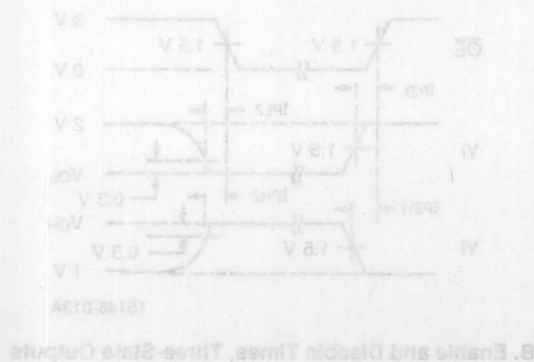
**Note:**

1. C_L includes probe and jig capacitance.

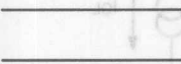


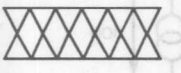
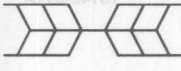
General Notes on Testing

Incoming test procedures on this device should be carefully planned, taking into account the complexity and power levels of the part. The following notes may be useful.

1. Ensure the part is adequately decoupled at the test head. Large changes in V_{CC} current as the device switches may cause erroneous function failures due to V_{CC} changes.
2. Do not leave inputs floating during any tests, as they may start to oscillate at high frequency.
3. Do not attempt to perform threshold tests at high speed. Following an input transition, ground current may change by as much as 400 mA in 5–8 ns. Inductance in the ground cable may allow the ground pin at the device to rise by hundreds of millivolts momentarily.
4. Use extreme care in defining input levels for AC tests. Many inputs may be changed at once, so there will be significant noise at the device pins and they may not actually reach V_{IL} or V_{IH} until the noise has settled. AMD recommends using $V_{IL} \leq 0$ V and $V_{IH} \geq 3$ V for AC tests.

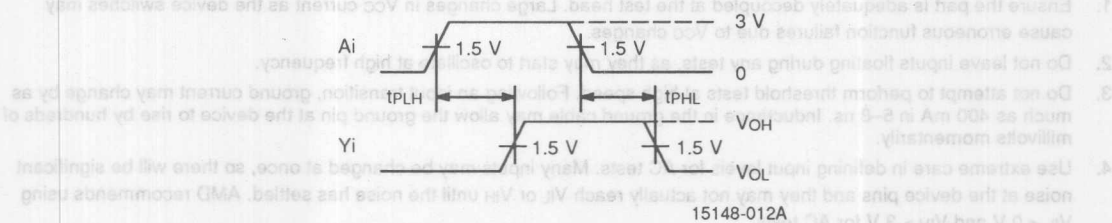


KEY TO SWITCHING WAVEFORMS

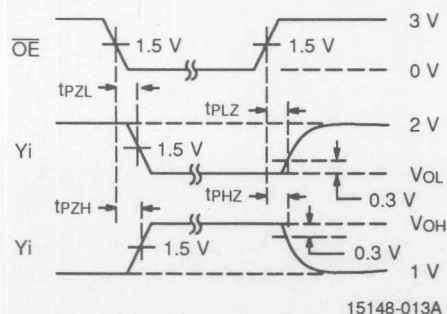
WAVEFORM	INPUTS	OUTPUTS
	Must Be Steady	Will Be Steady
	May Change from H to L	Will Be Changing from H to L
	May Change from L to H	Will Be Changing from L to H
	Don't Care, Any Change Permitted	Changing, State Unknown
	Does Not Apply	Center Line is High Impedance "Off" State

KS000010

SWITCHING TEST WAVEFORMS

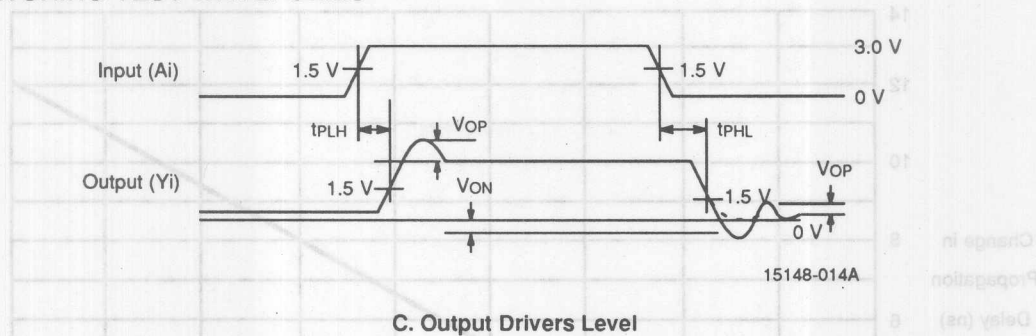


A. Propagation Delay Times

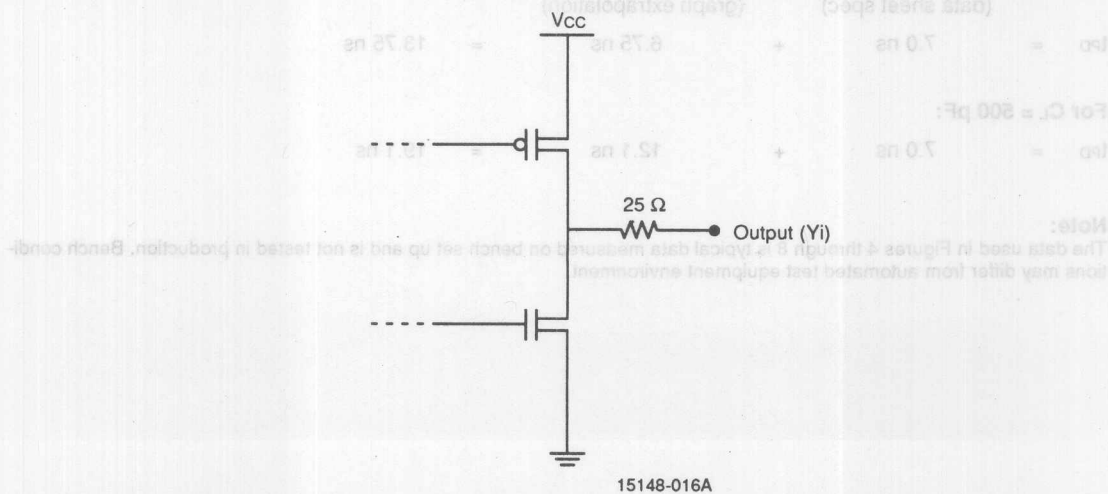
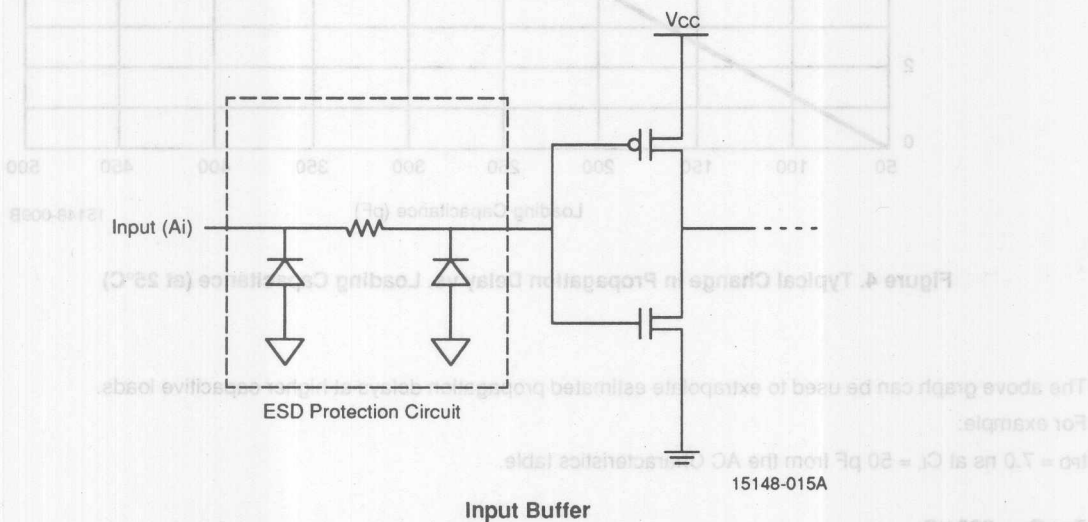


B. Enable and Disable Times, Three-State Outputs

SWITCHING TEST WAVEFORMS



EQUIVALENT INPUT/OUTPUT CIRCUIT DIAGRAMS



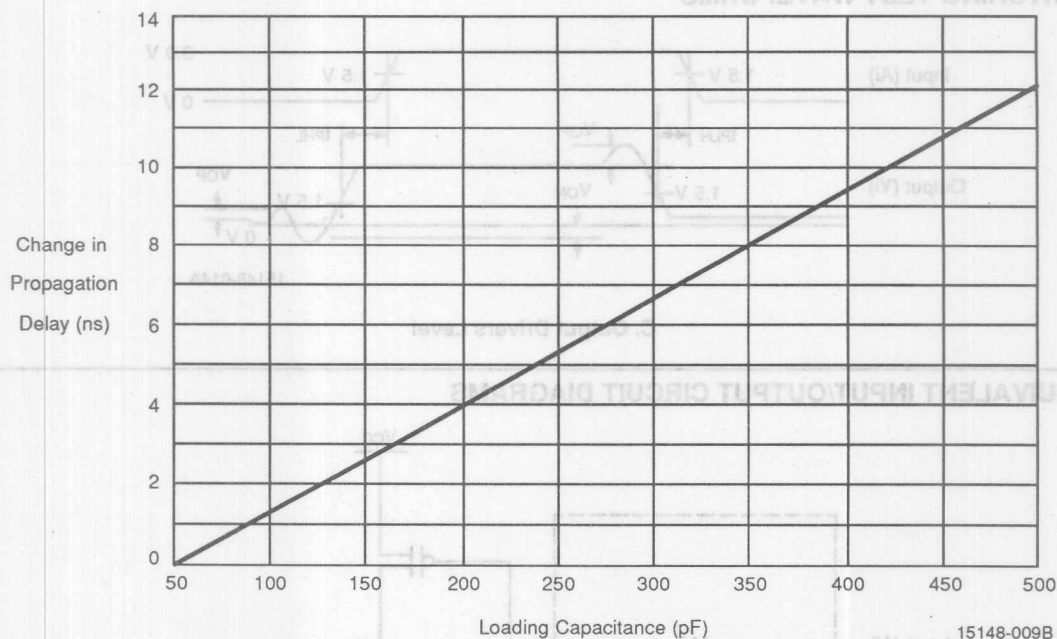


Figure 4. Typical Change In Propagation Delay vs. Loading Capacitance (at 25°C)

The above graph can be used to extrapolate estimated propagation delays at higher capacitive loads.

For example:

$t_{PD} = 7.0 \text{ ns}$ at $C_L = 50 \text{ pF}$ from the AC Characteristics table.

For $C_L = 300 \text{ pF}$:

	(data sheet spec)		(graph extrapolation)	
t_{PD}	=	7.0 ns	+	6.75 ns
				= 13.75 ns

For $C_L = 500 \text{ pF}$:

t_{PD}	=	7.0 ns	+	12.1 ns	=	19.1 ns
----------	---	--------	---	---------	---	---------

Note:

The data used in Figures 4 through 8 is typical data measured on bench set up and is not tested in production. Bench conditions may differ from automated test equipment environment.

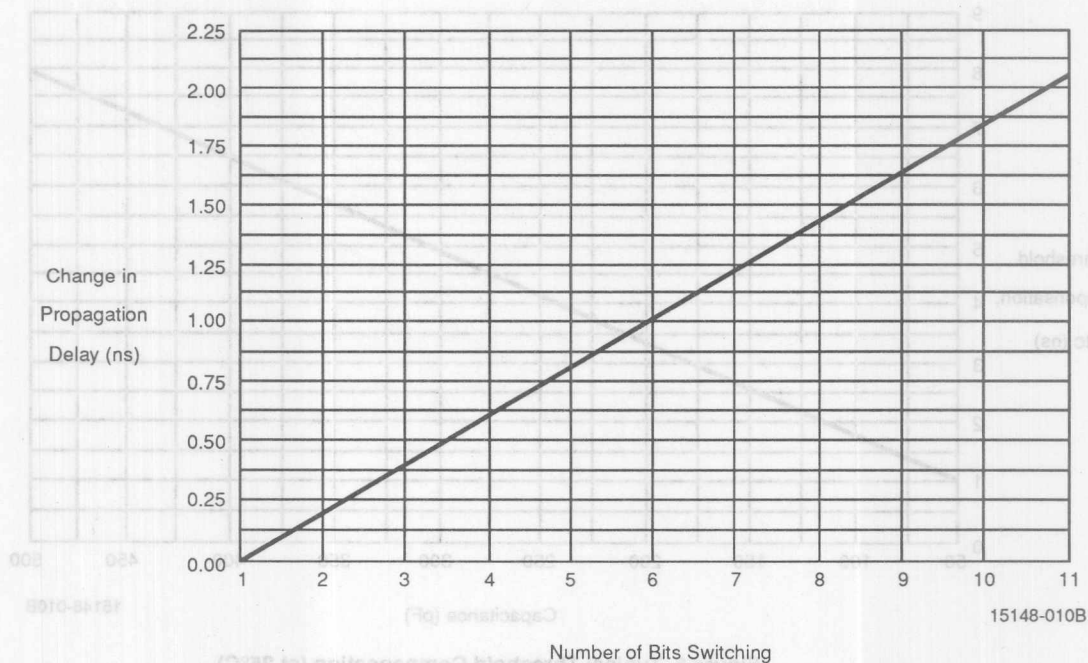
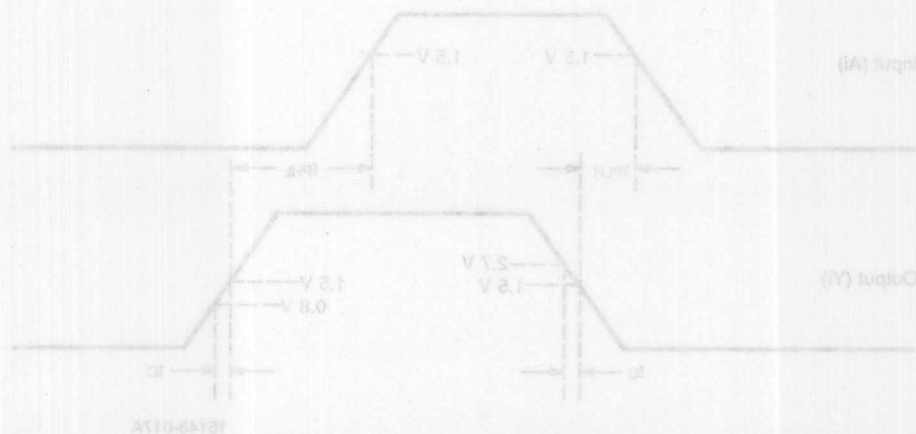


Figure 5. Typical Change in Propagation Delay vs. Number of Bits Switching (at 25°C, 50 pF)



It may be added to the specified propagation delays as a reference for system designers interested in estimating Am29C676 timing at TTL levels (0.8 V or 2.5 V).

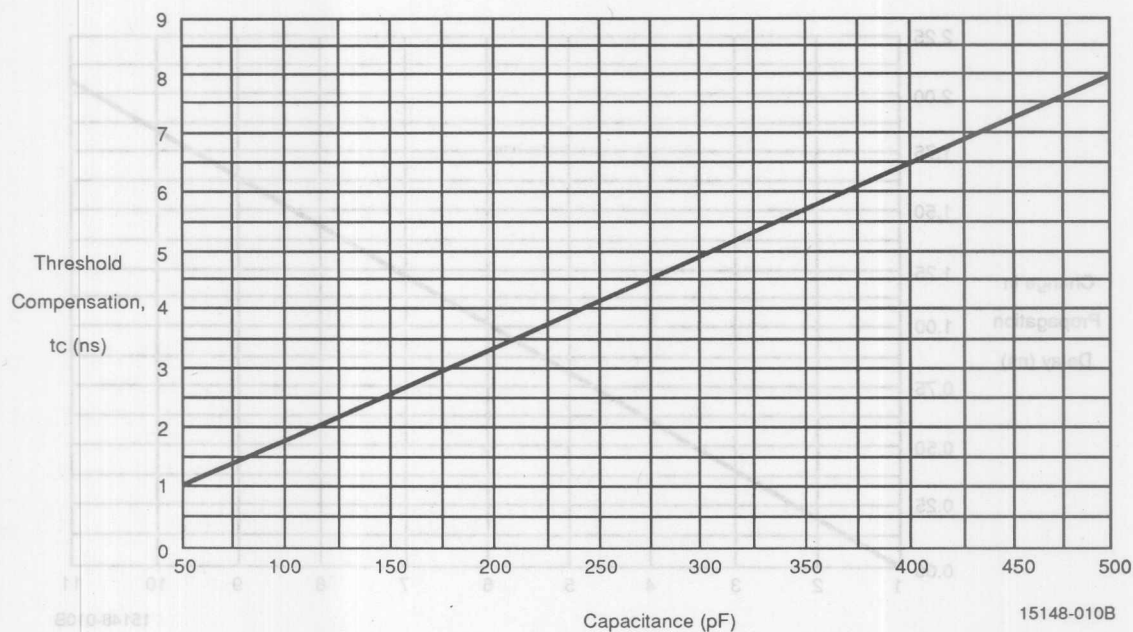
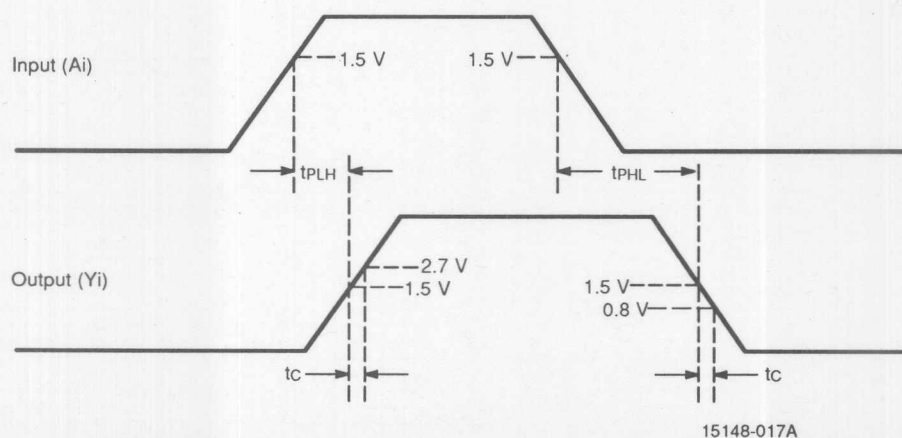


Figure 6. Typical Threshold Compensation (at 25°C)



t_c may be added to the specified propagation delays as a reference for system designers interested in estimating Am29C676 timing at TTL levels (0.8 V or 2.7 V).

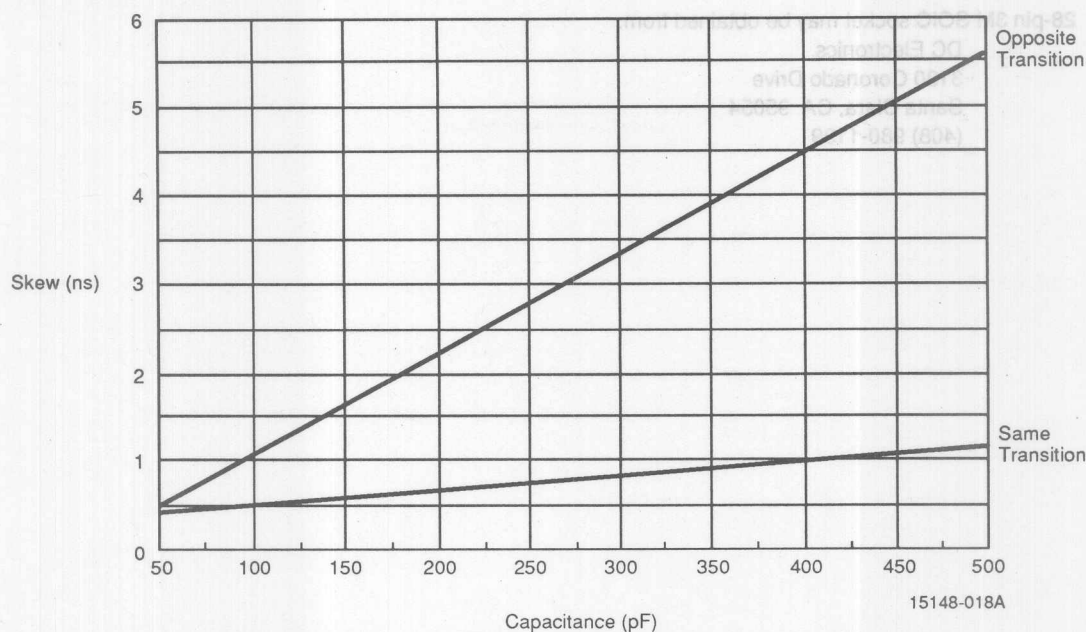


Figure 7. Typical Skew vs. Output Load Capacitance (at 25°C)

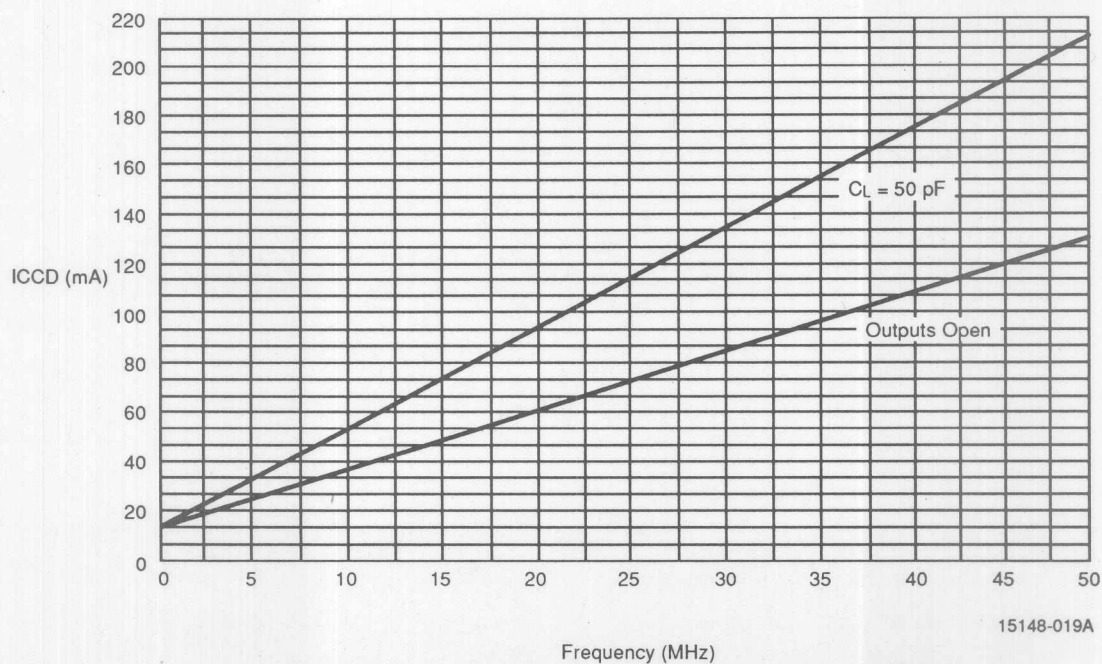
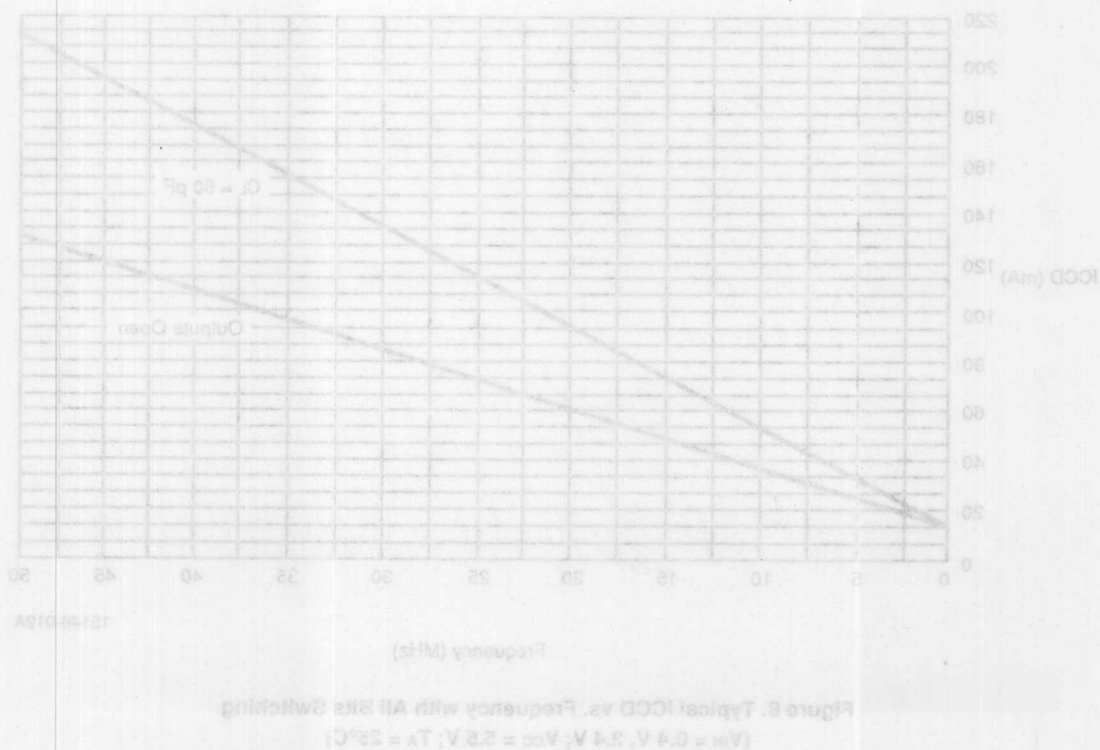
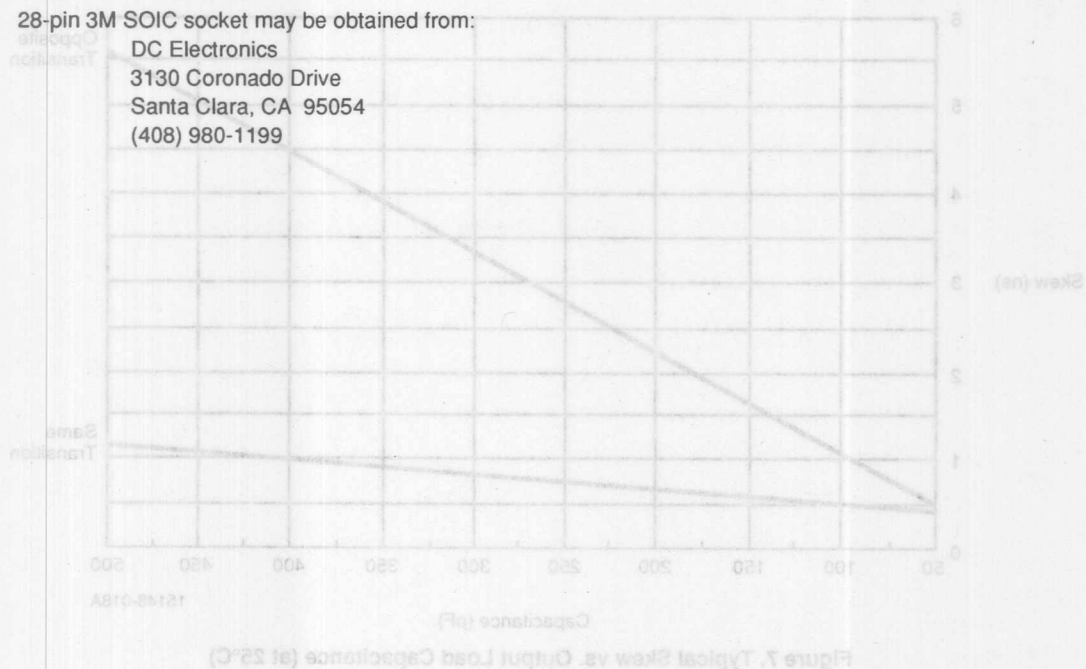


Figure 8. Typical ICCD vs. Frequency with All Bits Switching
($V_{IN} = 0.4\text{ V}, 2.4\text{ V}$; $V_{CC} = 5.5\text{ V}$; $T_A = 25^\circ\text{C}$)

SOIC Socket Information:

28-pin 3M SOIC socket may be obtained from:

DC Electronics
3130 Coronado Drive
Santa Clara, CA 95054
(408) 980-1199



Am29C827A/Am29C828A

High-Performance CMOS Bus Buffers

Advanced
Micro
Devices

DISTINCTIVE CHARACTERISTICS

- **High-speed CMOS buffers and inverters**
– D-Y delay = 4 ns typical
- **Low standby power**
- **JEDEC FCT-compatible specs**
- **Very high output drive**
– IOL = 48 mA Commercial, 32 mA Military
- **Extra-wide (10-bit) data paths**
- **200-mV typical hysteresis on data input ports**
- **Minimal speed degradation with multiple outputs switching**
- **Proprietary edge-rate controlled outputs**
dramatically reduce undershoots, overshoots, and ground bounce
- **Power-up/down disable circuit provides for glitch-free power supply sequencing**
- **Ideal for driving 1Mbit x 1 and 1Mbit x 4 DRAM address inputs**
- **Can be powered off while in 3-state, ideal for card edge interface applications**
- **JEDEC FCT-compatible specs**

GENERAL DESCRIPTION

The Am29C827A and Am29C828A CMOS Bus Buffers provide high-performance bus interface buffering for wide address/data paths or buses carrying parity. Both devices feature 10-bit wide data paths and NORed output enables for maximum control flexibility. The Am29C827A has non-inverting outputs, while the Am29C828A has inverting outputs. Each device has data inputs with 200-mV typical input hysteresis to provide improved noise immunity. The Am29C827A and Am29C828A are produced with AMD's exclusive CS11SA CMOS process, and feature typical propagation delays of 4 ns, as well as an output current drive of 48 mA.

The 29C827A and Am29C828A incorporate AMD's proprietary edge-controlled outputs in order to minimize simultaneous switching noise (ground bounce). By con-

trolling the output transient currents, ground bounce and output ringing have been greatly reduced. A modified AMD output provides a stable, usable voltage level in less time than a non-controlled output.

Additionally, speed degradation due to increasing number of outputs switching is reduced. Together, these benefits or edge-rate control result in significant increase in system performance despite a minor increase in device propagation delay.*

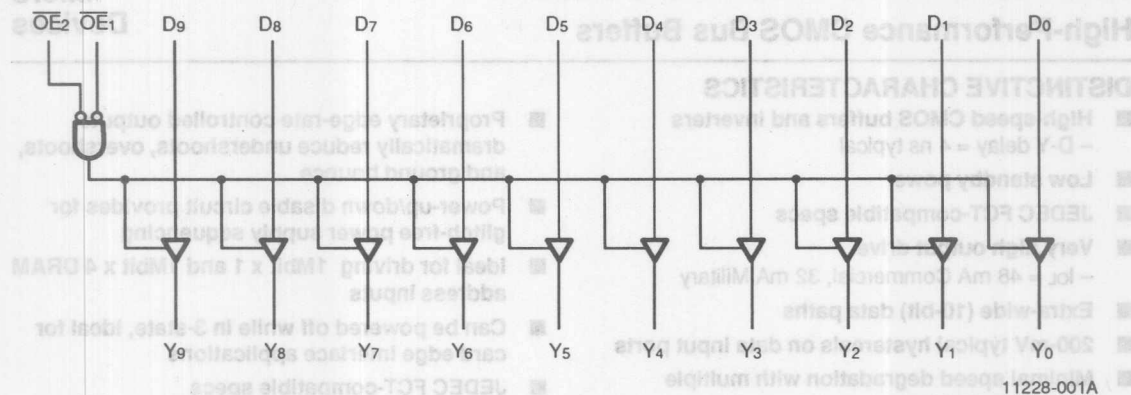
A unique I/O circuitry provides for high-impedance outputs during power-off and power-up/down sequencing, thus providing glitch-free operation for card-edge and other active bus applications.

The Am29C827A and Am29C828A are available in the standard package options: DIPs and SOICs.

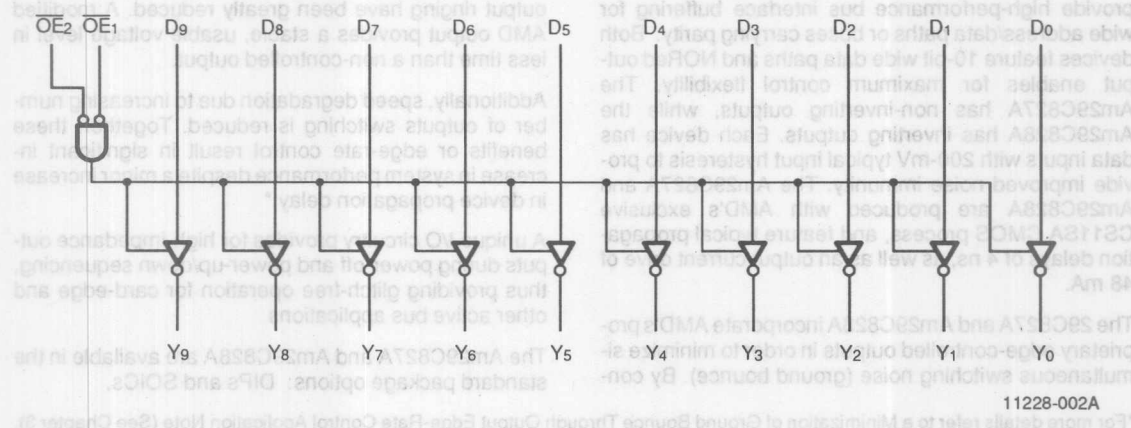
*For more details refer to a Minimization of Ground Bounce Through Output Edge-Rate Control Application Note (See Chapter 3).

BLOCK DIAGRAMS

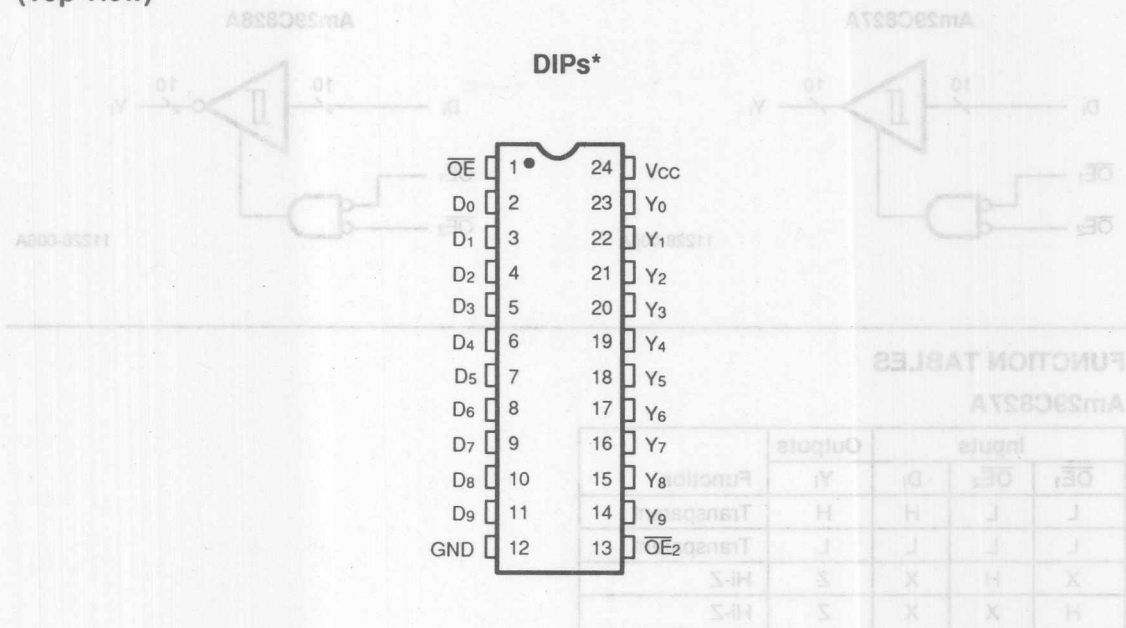
Am29C827A (Noninverting)



Am29C828A (Inverting)



CONNECTION DIAGRAMS (Top View)



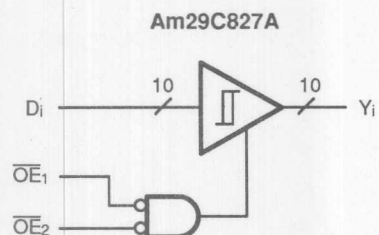
11228-003A

*Also available in Small Outline package; pinout identical to DIPs.

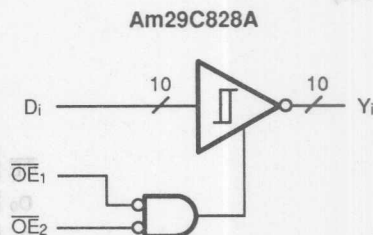
Inputs		Outputs	
OE ₁	OE ₂	D _i	Y _i
L	L	H	L
L	L	L	L
X	H	X	X
H	X	X	X

H = HIGH
L = LOW
X = Don't Care
Z = Hi-Z

LOGIC SYMBOLS



11228-005A



11228-006A

FUNCTION TABLES

Am29C827A

Inputs			Outputs	Function
\overline{OE}_1	\overline{OE}_2	D_i	Y_i	
L	L	H	H	Transparent
L	L	L	L	Transparent
X	H	X	Z	Hi-Z
H	X	X	Z	Hi-Z

Am29C828A

Inputs			Outputs	Function
\overline{OE}_1	\overline{OE}_2	D_i	Y_i	
L	L	H	L	Transparent
L	L	L	H	Transparent
X	H	X	Z	Hi-Z
H	X	X	Z	Hi-Z

H = HIGH
L = LOW
X = Don't Care
Z = Hi-Z

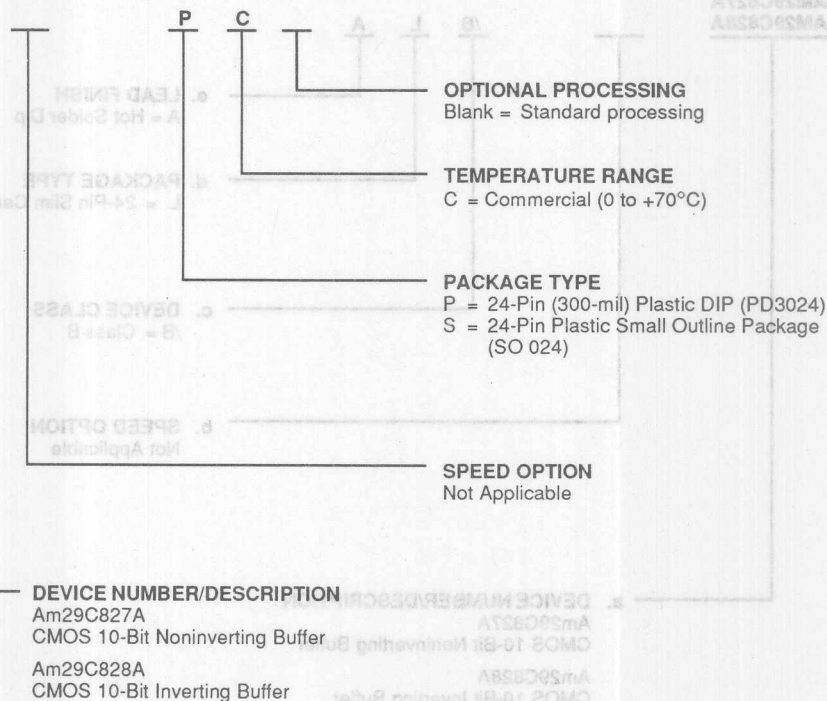
ORDERING INFORMATION

Standard Products

AMD standard products are available in several packages and operating ranges. The order number (Valid Combination) is formed by a combination of:

Device Number
Speed Option (if applicable)
Package Type
Temperature Range
Optional Processing

AM29C827A
AM29C828A



Valid Combinations	
AM29C827A	PC, SC
AM29C828A	

Valid Combinations

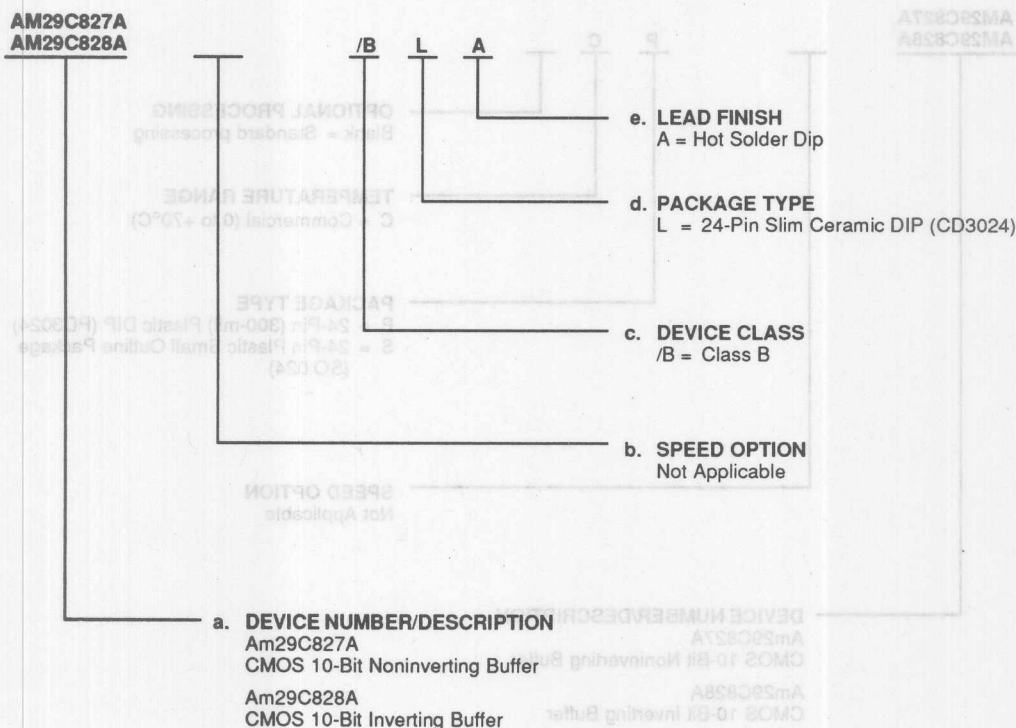
Valid Combinations list configurations planned to be supported in volume for this device. Consult the local AMD sales office to confirm availability of specific valid combinations or to check on newly released combinations, and to obtain additional data on AMD's standard military grade products.

MILITARY ORDERING INFORMATION

APL Products

AMD products for Aerospace and Defense applications are available in several packages and operating ranges. APL (Approved Products List) products are fully compliant with MIL-STD-883C requirements. The order number (Valid Combination) is formed by a combination of:

- Device Number
- Speed Option (if applicable)
- Device Class
- Package Type
- Lead Finish



Valid Combinations	
AM29C827A	/BLA
AM29C828A	

Valid Combinations

Valid Combinations list configurations planned to be supported in volume for this device. Consult the local AMD sales office to confirm availability of specific valid combinations, or to check on newly released combinations.

Group A Tests

Group A tests consist of Subgroups 1, 2, 3, 7, 8, 9, 10, 11.

PIN DESCRIPTION

\overline{OE}_i

Output Enables (Input, Active LOW)

When the \overline{OE}_1 and \overline{OE}_2 are both LOW, the outputs are enabled. When either one or both are HIGH, the outputs are in the Hi-Z state.

D_i

Data Inputs (Input)

D_i are the 10-bit data inputs.

ABSOLUTE MAXIMUM RATINGS

Storage Temperature

Data Outputs (Output)

Y_i are the 10-bit data outputs.

Supply Voltage to Ground Pin

DC Output Voltage

DC Input Voltage

DC Output Diode Current:

Into Output

Out of Output

DC Input Diode Current:

Into Input

Out of Input

DC Output Current per Pin:

Into Output

Out of Output

Total DC Ground Current

(n x IOL + m x IOL) mA (Note 1)

Total DC VCC Current

(n x IOH + m x IOH) mA (Note 1)

Note:

1. n = number of outputs, m = number of inputs

Stresses above those listed under Absolute Maximum Ratings may cause permanent device failure. Functionality at or above these limits is not implied. Exposure to absolute maximum ratings for extended periods may affect device reliability.

ABSOLUTE MAXIMUM RATINGS

Storage Temperature	-65 to +150°C
Supply Voltage to Ground Potential	
Continuous	-0.5 V to +7.0 V
DC Output Voltage	-0.5 V to +6.0 V
DC Input Voltage	-0.5 V to +6.0 V
DC Output Diode Current:	
Into Output	+50 mA
Out of Output	-50 mA
DC Input Diode Current:	
Into Input	+20 mA
Out of Input	-20 mA
DC Output Current per Pin:	
Into Output	+100 mA
Out of Output	-100 mA
Total DC Ground Current	
($n \times I_{OL} + m \times I_{CCT}$) mA (Note 1)	
Total DC V _{CC} Current	
($n \times I_{OH} + m \times I_{CCT}$) mA (Note 1)	

Note:

1. n = number of outputs, m = number of inputs.

Stresses above those listed under Absolute Maximum Ratings may cause permanent device failure. Functionality at or above these limits is not implied. Exposure to absolute maximum ratings for extended periods may affect device reliability.

OPERATING RANGES**Commercial (C) Devices**

Temperature (T _A)	0 to +70°C
Supply Voltage (V _{CC})	+4.5 V to +5.5 V

Military (M) Devices

Temperature (T _A)	-55 to +125°C
Supply Voltage (V _{CC})	+4.5 V to +5.5 V

Operating ranges define those limits between which the functionality of the device is guaranteed.

DC CHARACTERISTICS over operating ranges unless otherwise specified (for APL Products, Group A, Subgroups 1, 2, 3 are tested unless otherwise noted)

Parameter Symbol	Parameter Description	Test Conditions	Min.	Max.	Unit
V_{OH}	Output HIGH Voltage	$V_{CC} = 4.5\text{ V}$ $V_{IN} = V_{IH}\text{ or }V_{IL}$	$I_{OH} = -15\text{ mA}$	2.4	V
V_{OL}	Output LOW Voltage	$V_{CC} = 4.5\text{ V}$ $V_{IN} = V_{IH}\text{ or }V_{IL}$	MIL $I_{OL} = 32\text{ mA}$ COM'L $I_{OL} = 48\text{ mA}$	0.5 0.5	V V
V_{IH}	Input HIGH Voltage	Guaranteed Input Logical HIGH Voltage for All Inputs (Note 1)	2.0		V
V_{IL}	Input LOW Voltage	Guaranteed Input Logical LOW Voltage for All Inputs (Note 1)		0.8	V
V_I	Input Clamp Voltage	$V_{CC} = 4.5\text{ V}$, $I_{IN} = -18\text{ mA}$		-1.2	V
I_{IL}	Input LOW Current	$V_{CC} = 5.5\text{ V}$, $V_{IN} = \text{GND}$		-5	μA
I_{IH}	Input HIGH Current	$V_{CC} = 5.5\text{ V}$, $V_{IN} = 5.5\text{ V}$		5	μA
I_{OZH}	Output Off-State Current (High Impedance)	$V_{CC} = 5.5\text{ V}$, $V_O = 5.5\text{ V}$		+10	μA
I_{OZL}		$V_{CC} = 5.5\text{ V}$, $V_O = \text{or GND}$		-10	μA
I_{SC}	Output Short-Circuit Current	$V_{CC} = 5.5\text{ V}$, $V_O = 0\text{ V}$ (Note 2)	-60		mA
I_{CCQ}	Static Supply Current	$V_{CC} = 5.5\text{ V}$ Outputs Open	$V_{IN} = V_{CC}\text{ or GND}$	MIL 1.5 COM'L 1.2	mA
I_{CCT}			$V_{IN} = 3.4\text{ V}$	Data Input 1.5 $\overline{OE}_1, \overline{OE}_2$ 3.0	mA/ Bit
I_{CCD}^\dagger	Dynamic Supply Current	$V_{CC} = 5.5\text{ V}$ (Note 3)	Outputs Open	275	$\mu\text{A}/\text{MHz}/\text{Bit}$
			Outputs Loaded	400	

Notes:

1. Input thresholds are tested in combination with other DC parameters or by correlation.
2. Not more than one output shorted at a time. Duration should not exceed 100 milliseconds.
3. Measured at a frequency $\leq 10\text{ MHz}$ with 50% duty cycle.

† Not included in Group A tests.

SWITCHING CHARACTERISTICS for light capacitive loading over operating ranges unless otherwise specified (for APL Products, Group A, Subgroups 9, 10, 11 are tested unless otherwise noted)

Symbol	Parameter Description	Test Conditions*	Commercial		Military		Unit
			Min.	Max.	Min.	Max.	
t _{PLH}	Data (D _i) to Output (Y _i)	C _L = 50 pF R ₁ = 500 Ω R ₂ = 500 Ω	1.0	7.5	1.0	8.5	ns
t _{PHL}	Am29C827A (Noninverting) (Note 1)		1.0	7.5	1.0	8.5	ns
t _{PLH}	Data (D _i) to Output (Y _i)		1.0	7.5	0.5	8.5	ns
t _{PHL}	Am29C828A (Inverting) (Note 1)		1.0	7.5	0.5	8.5	ns
t _{ZH}	Output Enable Time \overline{OE} to Y _i		1.0	9	1.0	11	ns
t _{ZL}			3.0	12	3.0	14	ns
t _{HZ}	Output Disable Time \overline{OE} to Y _i		2.0	8	2.0	9	ns
t _{LZ}			3.0	8	2.0	9	ns

SWITCHING CHARACTERISTICS for heavy capacitive loading over operating ranges unless otherwise specified

Symbol	Parameter Description (Note 2)	Test Conditions*	Commercial		Military		Unit
			Min.	Max.	Min.	Max.	
tPLH	Data (Di) to Output (Yi)	CL = 300 pF R1 = 500 Ω R2 = 500 Ω	1.0	15.5	1.0	17.0	ns
tPHL	Am29C827A (Noninverting) (Note 1)		1.0	15.5	1.0	17.0	ns
tPLH	Data (Di) to Output (Yi)		1.0	13.5	0.5	15.0	ns
tPHL	Am29C828A (Inverting) (Note 1)		1.0	14	0.5	15.0	ns
tZH	Output Enable Time OE to Yi		1.0	13.5	1.0	15.0	ns
tZL			3.0	17	3.0	18.0	ns
tHZ	Output Disable Time OE to Yi	CL = 5 pF R1 = 500 Ω R2 = 500 Ω	2.0	7	2.0	8	ns
tLZ			3.0	7	2.0	8	ns

*See Test Circuit and Waveforms listed in Chapter 2.

Notes:

- For more details refer to a Minimization of Ground Bounce Through Output Edge-Rate Control Application Note (See Chapter 3).
- These parameters are guaranteed by characterization but not production tested.

Am29C983/Am29C983A

9-Bit x 4-Port Multiple Bus Exchange



DISTINCTIVE CHARACTERISTICS

- **Four bidirectional I/O ports with latches**
 - Replaces several bidirectional latches and transceivers
 - Permits multiple bus communication
 - Allows two independent communication channels
 - TTL compatibility
- **9 bit-wide ports to handle byte parity**
- **Two selection inputs per port**
 - Independent port interconnect control
 - Increased flexibility in data routing
- **Matched port decoding**
 - Simplifies external decode logic
 - Easily cascadable for wider buses
- **Latches for incoming and outgoing data**
 - Independent controls permit selective data capture
 - Ideal for stored operation
 - Readback feature for system diagnostics
- **Glitch-free outputs during power-up/down**
 - No power-up sequencing needed
 - Ideal for card-edge interface
- **48 mA output drive**
 - High-capacitance bus driving
- **High-performance CMOS**
 - Low stand-by power consumption
- **Two speeds available**
 - Am29C983
 - 9 ns (typ) port-to-port delay
 - 10 ns (typ) select-to-port delay
 - Am29C983A
 - 6 ns (typ) port-to-port delay
 - 7 ns (typ) select-to-port delay
- **Available in 68-pin PLCC package and 80-pin PQFP package for commercial applications**
- **Available in 68-pin PGA package for military applications (Am29C983 only)**

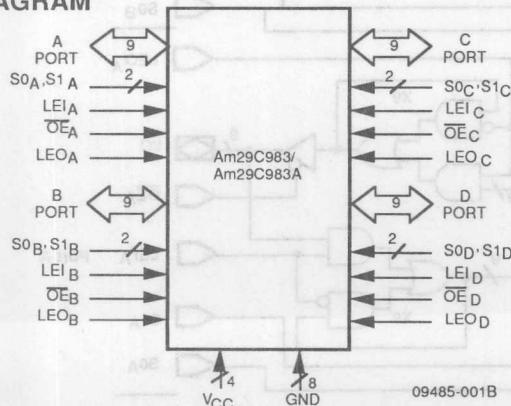
GENERAL DESCRIPTION

The Am29C983/A is a high-speed Multiple Bus Exchange device. It is organized as four 9-bit wide TTL-compatible I/O ports with Output Enable control for each port. Any port can serve either as a source (Input) port or as a destination (Output) port. When the output drivers of a port are disabled (high-impedance state), the port serves as a source port. When the drivers are enabled, the port serves as a destination port. Source port selection is made by two independent Select inputs at each port. This organization offers flexibility in implementing the Am29C983/A as a digital cross-point switch for multiple bus communication in a multiprocessing environment.

Each I/O port has an input latch to capture incoming data and an output latch to capture outgoing data. All input and output latches are independently controlled by active-HIGH Latch Enable inputs. This feature can be used to perform stored operation for byte-word compression and expansion to communicate between buses of different widths.

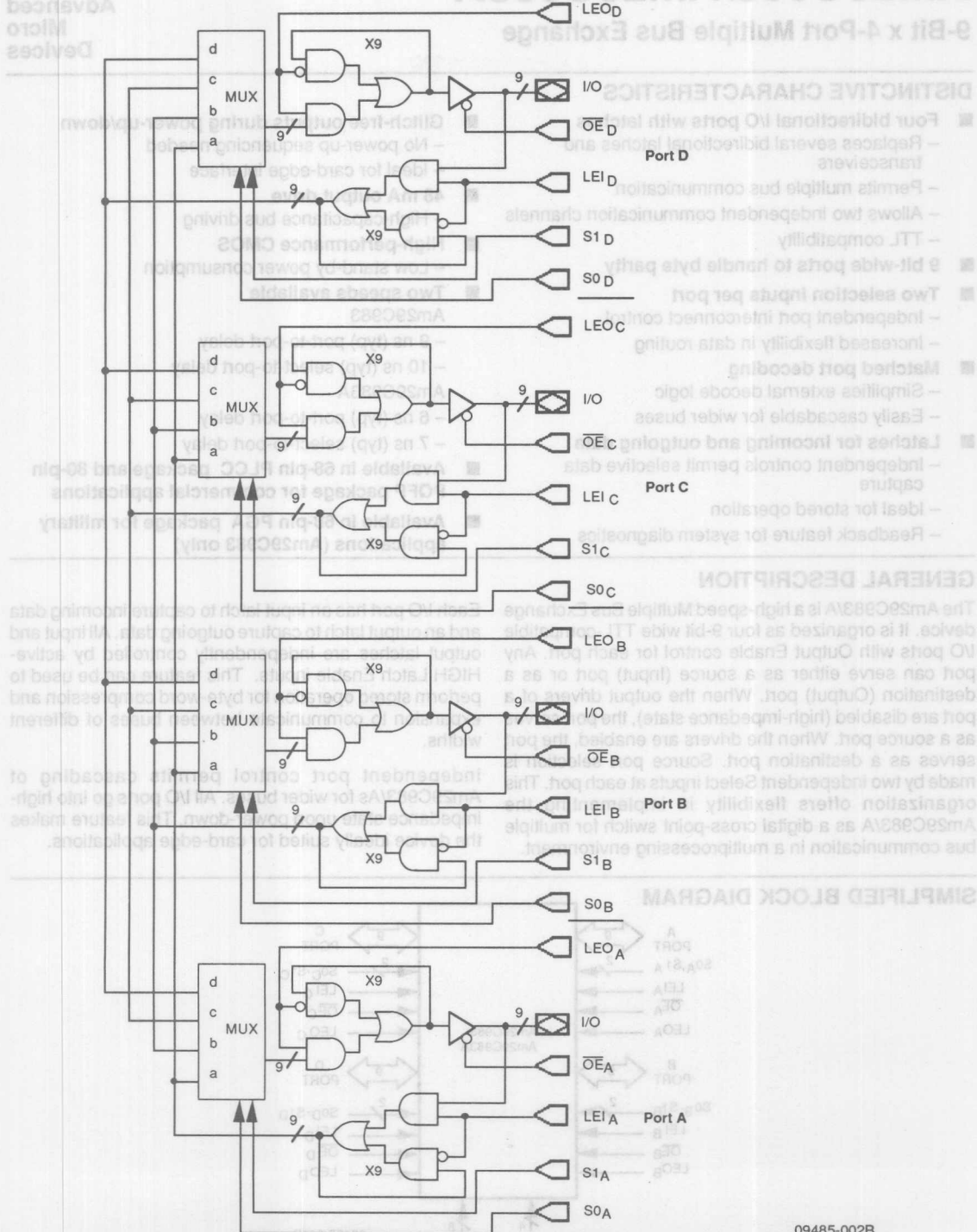
Independent port control permits cascading of Am29C983/As for wider buses. All I/O ports go into high-impedance state upon power-down. This feature makes the device ideally suited for card-edge applications.

SIMPLIFIED BLOCK DIAGRAM



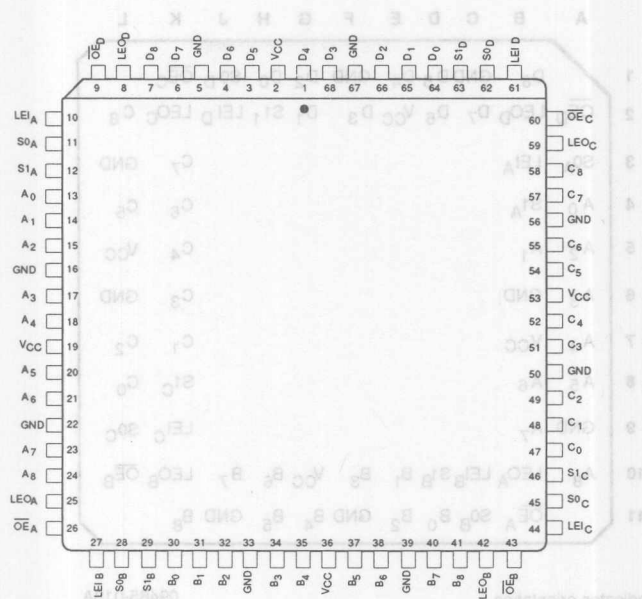
DETAILED BLOCK DIAGRAM

Advanced
Micro
Devices



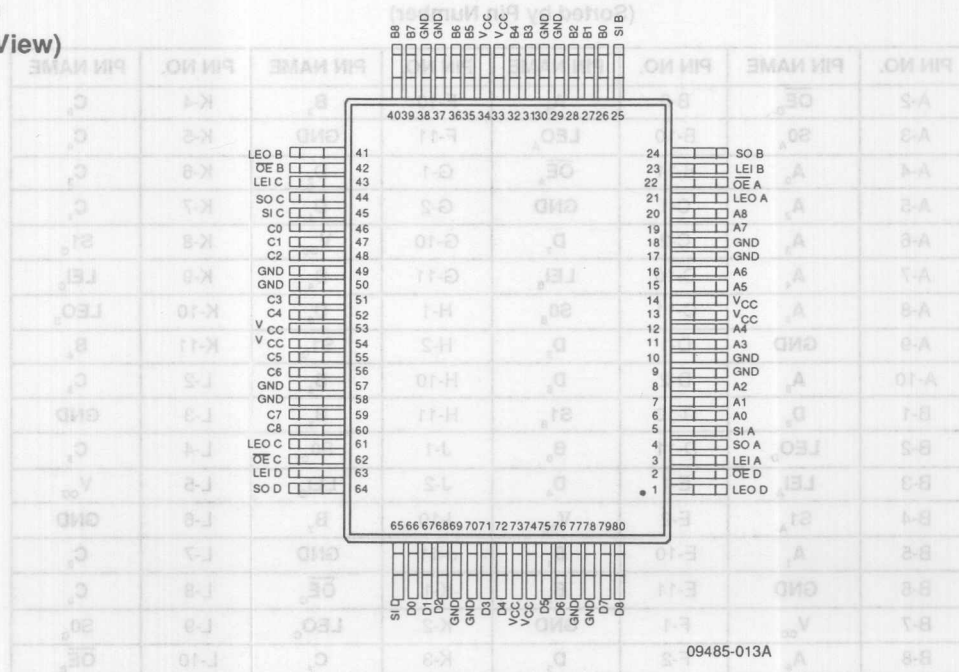
09485-002B

CONNECTION DIAGRAMS PLCC (Top View)

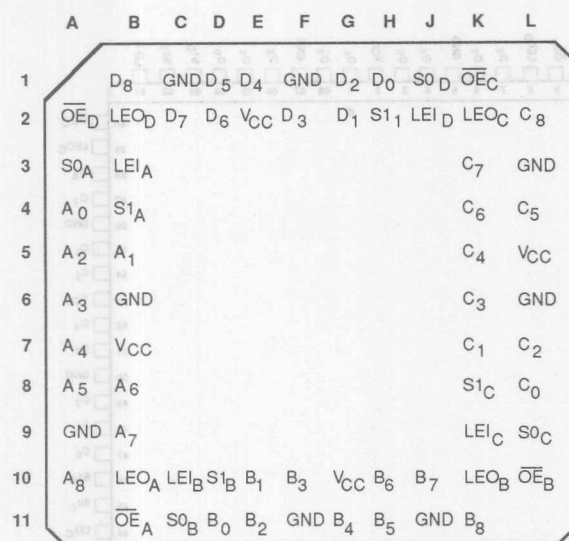


Note:
Pin 1 is marked for orientation.

PQFP (Top View)



PGA
(Bottom View)



Note: Notch indicates orientation.

09485-013A

PIN DESIGNATIONS
(Sorted by Pin Number)

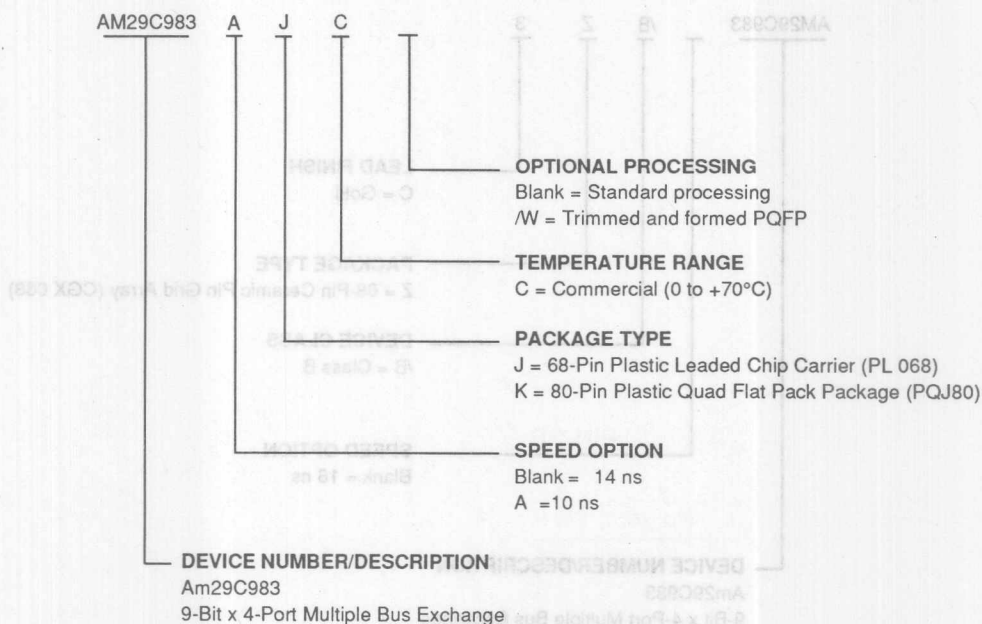
PIN NO.	PIN NAME	PIN NO.	PIN NAME	PIN NO.	PIN NAME	PIN NO.	PIN NAME
A-2	\overline{OE}_D	B-9	A_7	F-10	B_3	K-4	C_6
A-3	$S0_A$	B-10	LEO_A	F-11	GND	K-5	C_4
A-4	A_0	B-11	\overline{OE}_A	G-1	D_2	K-6	C_3
A-5	A_2	C-1	GND	G-2	D_1	K-7	C_1
A-6	A_3	C-2	D_7	G-10	V_{CC}	K-8	$S1_C$
A-7	A_4	C-10	LEI_B	G-11	B_4	K-9	LEI_C
A-8	A_5	C-11	$S0_B$	H-1	D_0	K-10	LEO_B
A-9	GND	D-1	D_5	H-2	$S1_D$	K-11	B_8
A-10	A_8	D-2	D_6	H-10	B_6	L-2	C_8
B-1	D_8	D-10	$S1_B$	H-11	B_5	L-3	GND
B-2	LEO_D	D-11	B_0	J-1	$S0_D$	L-4	C_5
B-3	LEI_A	E-1	D_4	J-2	LEI_D	L-5	V_{CC}
B-4	$S1_A$	E-2	V_{CC}	J-10	B_7	L-6	GND
B-5	A_1	E-10	B_1	J-11	GND	L-7	C_2
B-6	GND	E-11	B_2	K-1	\overline{OE}_C	L-8	C_0
B-7	V_{CC}	F-1	GND	K-2	LEO_C	L-9	$S0_C$
B-8	A_6	F-2	D_3	K-3	C_7	L-10	\overline{OE}_B

ORDERING INFORMATION

Standard Products

AMD standard products are available in several packages and operating ranges. The order number (Valid Combination) is formed by a combination of:

Device Number
Speed Option (if applicable)
Package Type
Temperature Range
Optional Processing



Valid Combinations

AM29C983	JC
AM29C983A	JC, KC/W

Valid Combinations

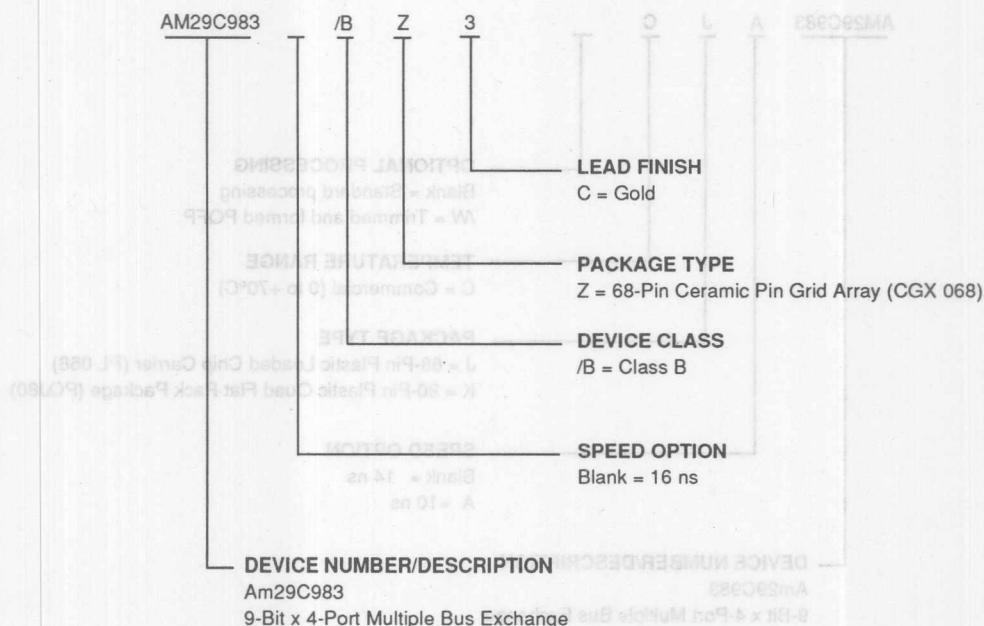
Valid Combinations list configurations planned to be supported in volume for this device. Consult the local AMD sales office to confirm availability of specific valid combinations, to check on newly released combinations, and to obtain additional data on AMD's standard military grade products.

ORDERING INFORMATION

APL Products

AMD products for Aerospace and Defense applications are available in several packages and operating ranges. APL (Approved Products List) products are fully compliant with MIL-STD-883C requirements. The order number (Valid Combination) is formed by a combination of:

Device Number
Speed Option (if applicable)
Package Type
Temperature Range
Optional Processing



Valid Combinations	
AM29C983	BZC

Valid Combinations

Valid Combinations list configurations planned to be supported in volume for this device. Consult the local AMD sales office to confirm availability of specific valid combinations, to check on newly released combinations, and to obtain additional data on AMD's standard military grade products.

Group A Tests

Group A tests consists of Subgroups 1, 2, 3, 7, 8, 9, 10, 11.

ABSOLUTE MAXIMUM RATINGS

Supply Voltage (V_{CC})	-0.5 to 7.0 V
DC Input Diode Current	-20 mA
(I_{IK}) ($V_{IN} < 0$ V)	-20 mA
($V_{IN} > V_{CC}$ if applicable)	+20 mA
DC Input Voltage (V_{IN})	-0.5 to 7 V
DC Output Diode Current	-50 mA
(I_{OK}) ($V_{OUT} < 0$ V)	-50 mA
($V_{OUT} > V_{CC}$ if applicable)	+50 mA
DC Output Current per Output Pin:	
I_{SINK}	+70 mA
I_{SOURCE}	-30 mA
DC Output Voltage (V_{OUT})	-0.5 to 7 V
Total DC Ground Current (I_{GND})	1750 mA
Total DC V_{CC} Current (I_{CC})	575 mA
Storage Temperature	-65 to +150°C

Stresses above those listed under ABSOLUTE MAXIMUM RATINGS may cause permanent device failure. Functionality at or above these limits is not implied. Exposure to absolute maximum ratings for extended periods may affect device reliability.

OPERATING RANGES

Commercial (C) Devices	0 to +70°C
Temperature (T_A)	+4.5 to +5.5 V
Supply Voltage (V_{CC})	
Operating ranges define those limits between which the functionality of the device is guaranteed.	

DC CHARACTERISTICS over operating range unless otherwise specified

Am29C983						
Parameter Symbol	Parameter Description	Test Conditions		Min	Max	Unit
V_{OH}	Output HIGH Voltage	$V_{CC} = 4.5$ V $V_{IN} = V_{IL}$ or V_{IH}	$I_{OH} = -15$ mA	2.4		V
V_{OL}	Output LOW Voltage	$V_{CC} = 4.5$ V $V_{IN} = V_{IL}$ or V_{IH}	$I_{OL} = 48$ mA		0.5	V
V_{IH}	Input HIGH Voltage	(Note 1)		2.0		V
V_{IL}	Input LOW Voltage	(Note 1)			0.8	V
V_{IC}	Input Clamp Voltage	$V_{CC} = 4.5$ V, $I_{IN} = -18$ mA			-1.2	V
I_{IL}	Input LOW Current (Select Inputs)	$V_{CC} = 5.5$ V, $V_{IN} = 0$ V			-10	μ A
I_{IH}	Input HIGH Current (Select Inputs)	$V_{CC} = 5.5$ V, $V_{IN} = 5.5$ V			10	μ A
I_{OZL}	Off-State Leakage Current (I/O Ports)	$V_{CC} = 5.5$ V, $V_{OUT} = 0$ V			-20	μ A
I_{OZH}	Off-State Leakage Current (I/O Ports)	$V_{CC} = 5.5$ V, $V_{OUT} = 5.5$ V			20	μ A
I_{SC}	Output Short-Circuit Current	$V_{CC} = 5.5$ V, $V_{OUT} = 0$ V (Note 2)		-60		mA
I_{CCO}	Quiescent Power Supply Current (Note 4)	$V_{CC} = 5.5$ V, $V_{IN} = 5.5$ V or GND Outputs Open			1.5	mA

DC CHARACTERISTICS (Continued)

Parameter Symbol	Parameter Description	Test Conditions	Min.	Max.	Unit
I_{CCT}	Power Supply Current TTL Input HIGH (Note 4)	$V_{CC} = 5.5 \text{ V}$, $V_{IN} = 2.4 \text{ V}$ Other Inputs at V_{CC} or GND		3.0	mA/ Input
I_{CCD}	Dynamic Power Supply Current (Note 4)	$V_{CC} = 5.5 \text{ V}$, Outputs Open One Output Toggling (Note 3)		500	$\mu\text{A/}$ MHz/Bit

Am29C983A

Parameter Symbol	Parameter Description	Test Conditions	Min.	Max.	Unit
V_{OH}	Output HIGH Voltage	$V_{CC} = 4.5 \text{ V}$ $V_{IN} = V_{IL}$ or V_{IH} $I_{OH} = -15 \text{ mA}$	2.4		V
V_{OL}	Output LOW Voltage	$V_{CC} = 4.5 \text{ V}$ $V_{IN} = V_{IL}$ or V_{IH} $I_{OL} = 48 \text{ mA}$		0.5	V
V_{IH}	Input HIGH Voltage	(Note 1)	2.0		V
V_{IL}	Input LOW Voltage	(Note 1)		0.8	V
V_{IC}	Input Clamp Voltage	$V_{CC} = 4.5 \text{ V}$, $I_{IN} = -18 \text{ mA}$		-1.2	V
I_{IL}	Input LOW Current (Select Inputs)	$V_{CC} = 5.5 \text{ V}$, $V_{IN} = 0 \text{ V}$		-10	μA
I_{IH}	Input HIGH Current (Select Inputs)	$V_{CC} = 5.5 \text{ V}$, $V_{IN} = 5.5 \text{ V}$		10	μA
I_{OZL}	Off-State Leakage Current (I/O Ports)	$V_{CC} = 5.5 \text{ V}$, $V_{OUT} = 0 \text{ V}$		-20	μA
I_{OZH}	Off-State Leakage Current (I/O Ports)	$V_{CC} = 5.5 \text{ V}$, $V_{OUT} = 5.5 \text{ V}$		20	μA
I_{SC}	Output Short-Circuit Current	$V_{CC} = 5.5 \text{ V}$, $V_{OUT} = 0 \text{ V}$ (Note 2)	-60		mA
I_{CCO}	Quiescent Power Supply Current (Note 4)	$V_{CC} = 5.5 \text{ V}$, $V_{IN} = 5.5 \text{ V}$ or GND Outputs Open		1.5	mA
I_{CCT}	Power Supply Current TTL Input HIGH (Note 4)	$V_{CC} = 5.5 \text{ V}$, $V_{IN} = 2.4 \text{ V}$ Other Inputs at V_{CC} or GND		3.0	mA/ Input
I_{CCD}	Dynamic Power Supply Current (Note 4)	$V_{CC} = 5.5 \text{ V}$, Outputs Open One Output Toggling (Note 3)		500	$\mu\text{A/}$ MHz/Bit

Notes:

- Input thresholds are tested in combination with other DC parameters or by correlation.
- Not more than one output shorted at a time. Duration of short-circuit test not to exceed 100 milliseconds.
- Measured at a frequency of < 10 MHz with 50% duty cycle. Unused inputs are at V_{CC} or GND.
- Calculation of total device I_{CC} : $I_{CC} = I_{CCO} + I_{CCT} \times M_T \times D_H + I_{CCD} \times ((C_L + 91) \times 91) \times f \times N$
Where
 C_L = Load Capacitance in pF per output
 f = Frequency in MHz
 N = Average number of outputs switching
 M_T = Number of inputs at logic HIGH
 D_H = Duty cycle for each input HIGH

SWITCHING CHARACTERISTICS over COMMERCIAL operating range unless otherwise specified

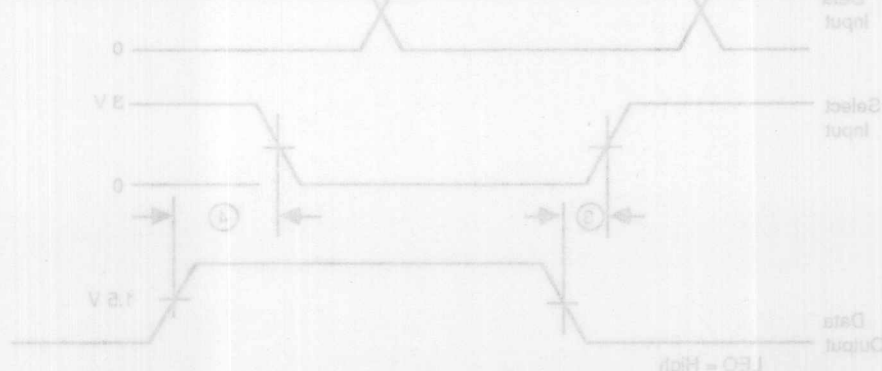
Am29C983			Test Conditions	Commercial		
No.	Parameter Symbol	Parameter Description		Min	Max	Unit
1	t_{PLH}	Propagation Delay Port to Port	$C_L = 50 \text{ pF}$ $R_1 = 500 \text{ Ohms}$ $R_2 = 500 \text{ Ohms}$	1.5	14	ns
2	t_{PHL}	LEI = HIGH, LEO = HIGH		1.5	14	ns
3	t_{PLH}	Propagation Delay Select Input to Port LEO = HIGH		1.5	18	ns
4	t_{PHL}			1.5	18	ns
5	t_{PLH}	Propagation Delay LEI to Port		1.5	18	ns
6	t_{PHL}	LEO = HIGH		1.5	18	ns
7	t_{PLH}	Propagation Delay LEO to Port		1.5	14	ns
8	t_{PHL}			1.5	14	ns
9	t_{PZH}	Output Enable Time \overline{OE} to Port		1	14	ns
10	t_{PZL}			1	14	ns
11	t_{PHZ}	Output Disable Time \overline{OE} to Port		0	12	ns
12	t_{PLZ}			0	12	ns
13	t_s	Port to LEI Setup		2		ns
14	t_h	Port to LEI Hold		3		ns
15	t_s	Port to LEO Setup		4.5		ns
16	t_h	Port to LEO Hold		1.5		ns
17	t_s	Select to LEO Setup		6		ns
18	t_h	Select to LEO Hold		0		ns
19	t_s	LEI to LEO Setup		6		ns
20	t_h	LEI to LEO Hold		0		ns
21	t_{PWH}	LEI, LEO Pulse Width HIGH		6		ns

SWITCHING CHARACTERISTICS over MILITARY operating range unless otherwise specified (for APL products, Group A, Subgroups 9, 10, 11 are tested unless otherwise noted)

Am29C983				Military		
No.	Parameter Symbol	Parameter Description	Test Conditions	Min	Max	Unit
1	t_{PLH}	Propagation Delay Port to Port	$C_L = 50 \text{ pF}$ $R_1 = 500 \text{ Ohms}$ $R_2 = 500 \text{ Ohms}$	1.0	16	ns
2	t_{PHL}	LEI = HIGH, LEO = HIGH		1.0	16	ns
3	t_{PLH}	Propagation Delay Select Input to Port LEO = HIGH		1.0	20	ns
4	t_{PHL}			1.0	20	ns
5	t_{PLH}	Propagation Delay LEI to Port LEO = HIGH		1.0	20	ns
6	t_{PHL}			1.0	20	ns
7	t_{PLH}	Propagation Delay LEO to Port		1.0	16	ns
8	t_{PHL}			1.0	16	ns
9	t_{PZH}	Output Enable Time OE to Port		1.5	16	ns
10	t_{PZL}			1.0	16	ns
11	t_{PHZ}	Output Disable Time OE to Port		0	14	ns
12	t_{PLZ}			0	14	ns
13	t_s	Port to LEI Setup		3		ns
14	t_h	Port to LEI Hold		4		ns
15	t_s	Port to LEO Setup		5.5		ns
16	t_h	Port to LEO Hold		2.5		ns
17	t_s	Select to LEO Setup		7		ns
18	t_h	Select to LEO Hold		1		ns
19	t_s	LEI to LEO Setup		7		ns
20	t_h	LEI to LEO Hold		1		ns
21	t_{PWH}	LEI, LEO Pulse Width HIGH		7		ns

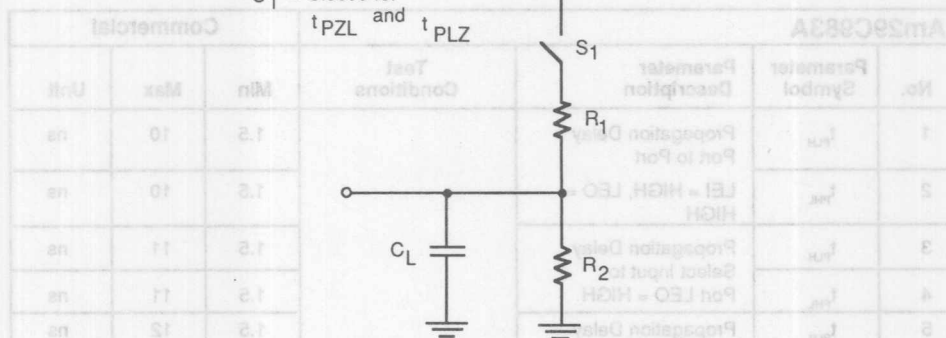
SWITCHING CHARACTERISTICS over COMMERCIAL operating range unless otherwise specified

Am29C983A				Commercial		
No.	Parameter Symbol	Parameter Description	Test Conditions	Min	Max	Unit
1	t_{PLH}	Propagation Delay Port to Port		1.5	10	ns
2	t_{PHL}	LEI = HIGH, LEO = HIGH		1.5	10	ns
3	t_{PLH}	Propagation Delay Select Input to Port LEO = HIGH		1.5	11	ns
4	t_{PHL}			1.5	11	ns
5	t_{PLH}	Propagation Delay LEI to Port		1.5	12	ns
6	t_{PHL}	LEO = HIGH		1.5	12	ns
7	t_{PLH}	Propagation Delay LEO to Port		1.5	10	ns
8	t_{PHL}			1.5	10	ns
9	t_{PZH}	Output Enable Time \overline{OE} to Port		1	10	ns
10	t_{PZL}			1	10	ns
11	t_{PHZ}	Output Disable Time \overline{OE} to Port		0	9	ns
12	t_{PLZ}			0	9	ns
13	t_s	Port to LEI Setup		2		ns
14	t_h	Port to LEI Hold		3		ns
15	t_s	Port to LEO Setup		4.5		ns
16	t_h	Port to LEO Hold		1.5		ns
17	t_s	Select to LEO Setup		6		ns
18	t_h	Select to LEO Hold		0		ns
19	t_s	LEI to LEO Setup		6		ns
20	t_h	LEI to LEO Hold		0		ns
21	t_{PWH}	LEI, LEO Pulse Width HIGH		6		ns



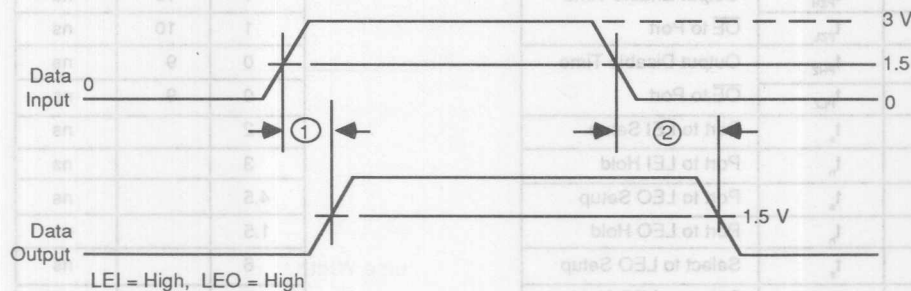
SWITCHING TEST CIRCUIT

S_1 = Open normally
 S_1 = Closed for t_{PZL} and t_{PLZ}



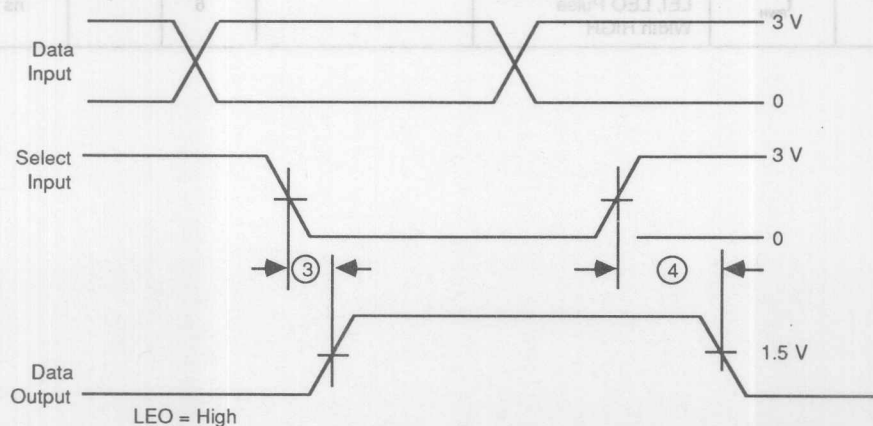
09485-007A

SWITCHING TEST WAVEFORMS



Propagation Delay—Port-to-Port

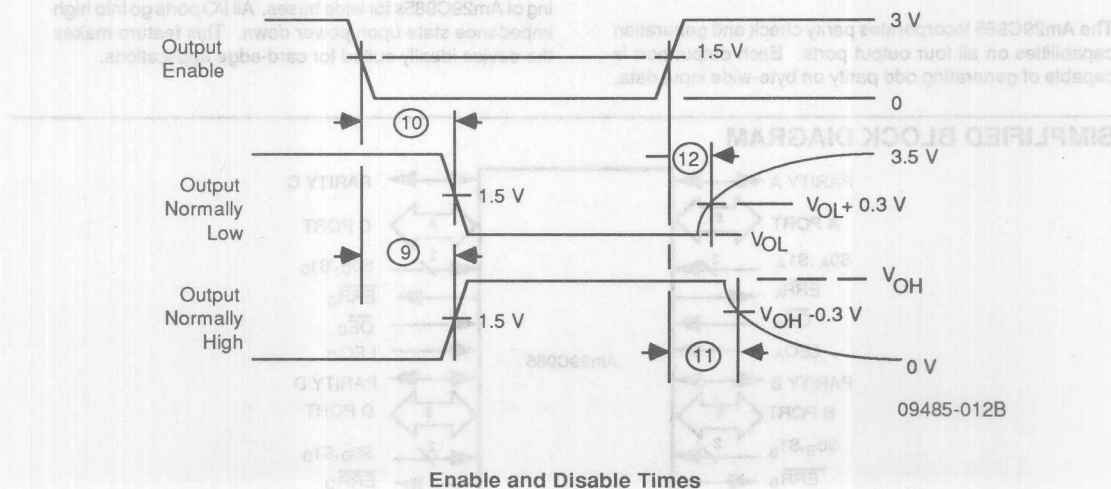
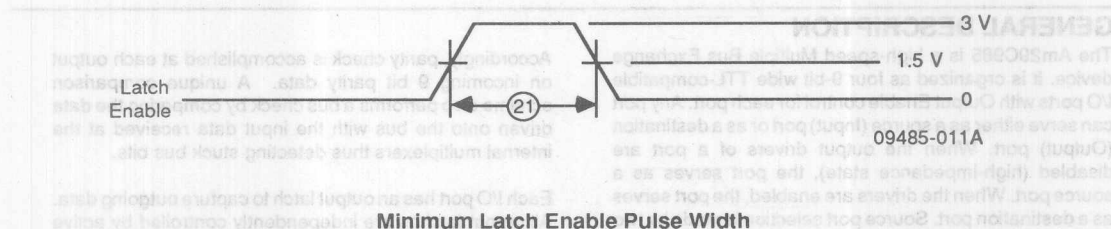
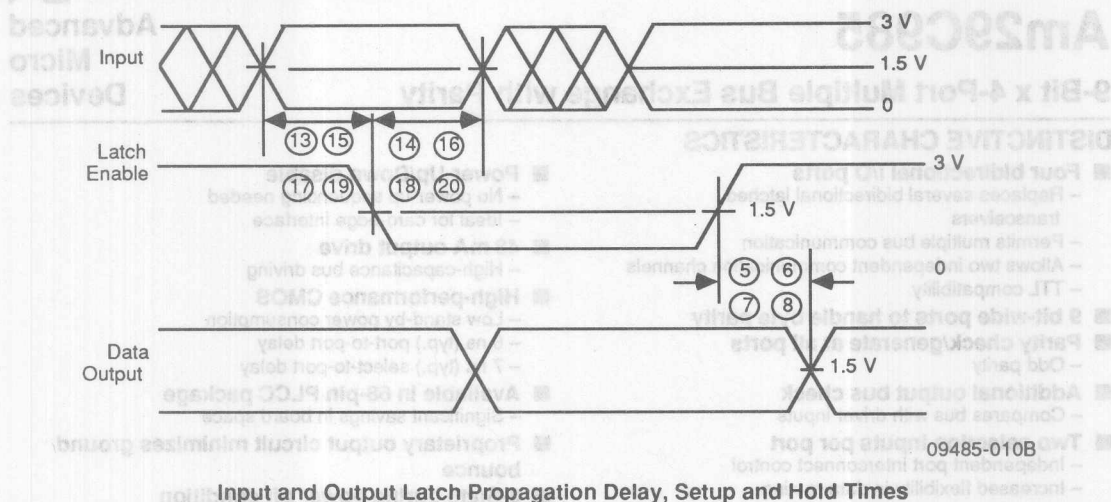
09485-008B



Propagation Delay—Select-to-Port

09485-009B

SWITCHING TEST WAVEFORMS



Am29C985

9-Bit x 4-Port Multiple Bus Exchange with Parity

Advanced
Micro
Devices

DISTINCTIVE CHARACTERISTICS

- **Four bidirectional I/O ports**
 - Replaces several bidirectional latched transceivers
 - Permits multiple bus communication
 - Allows two independent communication channels
 - TTL compatibility
- **9 bit-wide ports to handle byte parity**
- **Parity check/generate at all ports**
 - Odd parity
- **Additional output bus check**
 - Compares bus with driver inputs
- **Two selection inputs per port**
 - Independent port interconnect control
 - Increased flexibility in data routing
- **Matched port decoding**
 - Simplifies external decode logic
 - Easily cascadable for wider buses
- **Power-Up/Down disable**
 - No power-up sequencing needed
 - Ideal for card-edge interface
- **48 mA output drive**
 - High-capacitance bus driving
- **High-performance CMOS**
 - Low stand-by power consumption
 - 6 ns (typ.) port-to-port delay
 - 7 ns (typ.) select-to-port delay
- **Available in 68-pin PLCC package**
 - Significant savings in board space
- **Proprietary output circuit minimizes ground bounce**
- **3-State during power off condition**

GENERAL DESCRIPTION

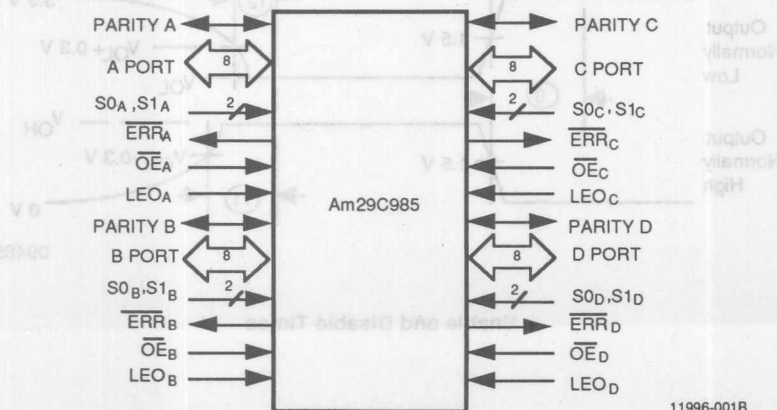
The Am29C985 is a high-speed Multiple Bus Exchange device. It is organized as four 9-bit wide TTL-compatible I/O ports with Output Enable control for each port. Any port can serve either as a source (Input) port or as a destination (Output) port. When the output drivers of a port are disabled (high-impedance state), the port serves as a source port. When the drivers are enabled, the port serves as a destination port. Source port selection is made by two independent Select inputs at each port. This organization offers flexibility in implementing the Am29C985 as a digital cross-point switch for multiple bus communication in a multiprocessing environment.

The Am29C985 incorporates parity check and generation capabilities on all four output ports. Each output port is capable of generating odd parity on byte-wide input data.

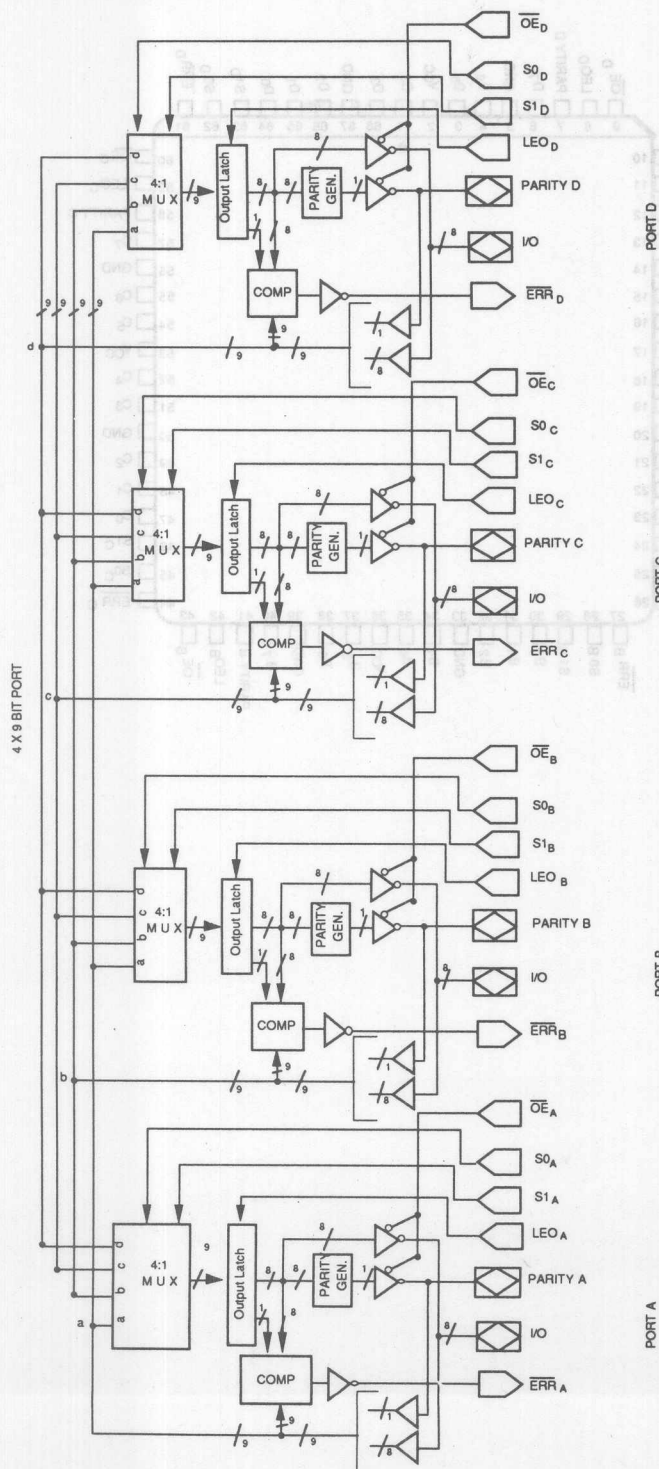
Accordingly, parity check is accomplished at each output on incoming 9 bit parity data. A unique comparison scheme also performs a bus check by comparing the data driven onto the bus with the input data received at the internal multiplexers thus detecting stuck bus bits.

Each I/O port has an output latch to capture outgoing data. All output latches are independently controlled by active HIGH Output Latch Enable inputs. This feature can be used to perform stored operation for byte-word compression and expansion to communicate between buses of different widths. Independent port control permits cascading of Am29C985s for wide buses. All I/O ports go into high impedance state upon power down. This feature makes the device ideally suited for card-edge applications.

SIMPLIFIED BLOCK DIAGRAM



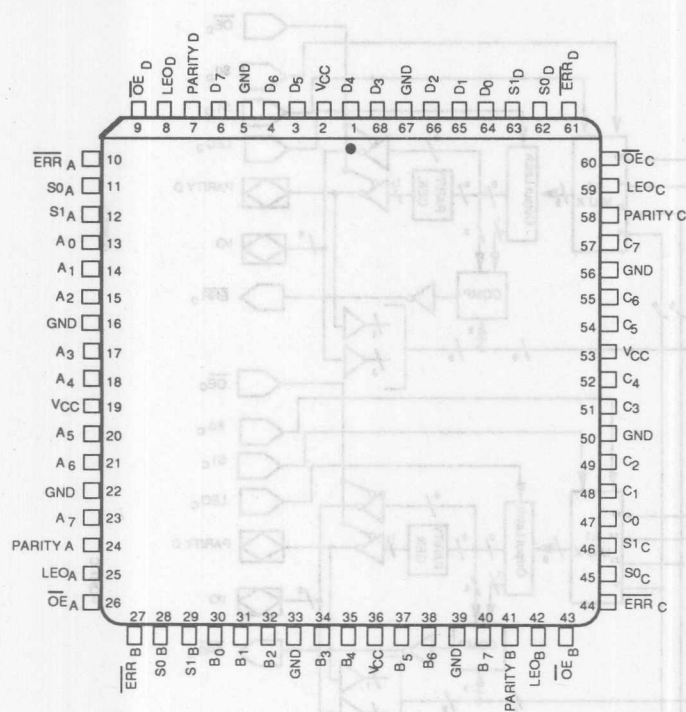
DETAILED BLOCK DIAGRAM



11996-002A

CONNECTION DIAGRAMS PLCC (Top View)

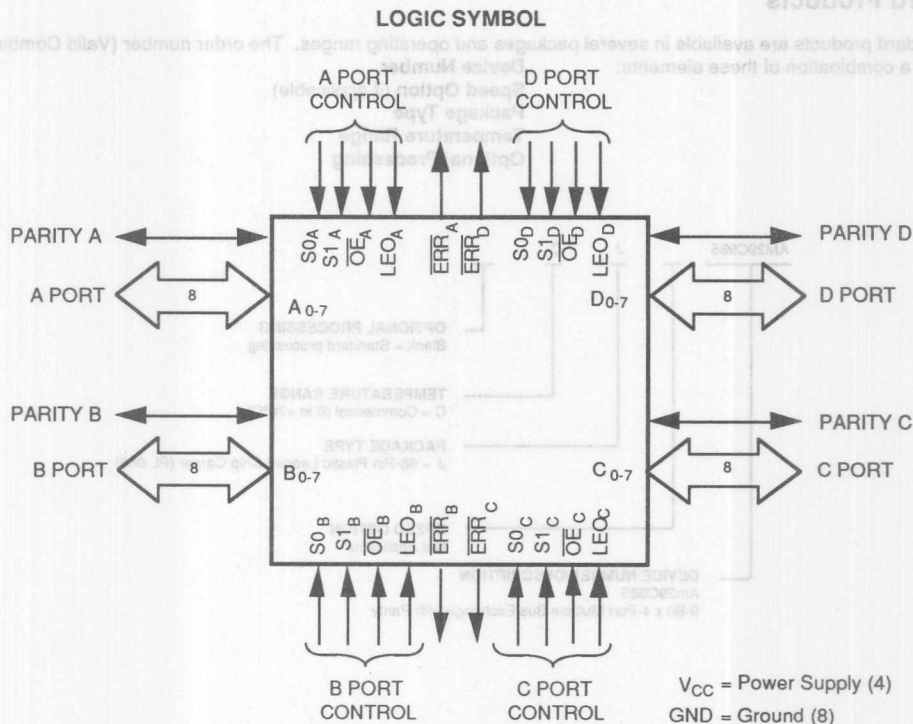
DETAILED BLOCK DIAGRAM



11996A-003A

Note:
Pin 1 is marked for orientation.

LOGIC SYMBOL



Valid Combinations for configurations planned to be supported in volume for this device. Consult the local AMD sales office to confirm availability of specific valid combinations. To check on newly released conditions, and to obtain additional data on AMD's standard military grade products.

Valid Combinations	
Am29C985	JC

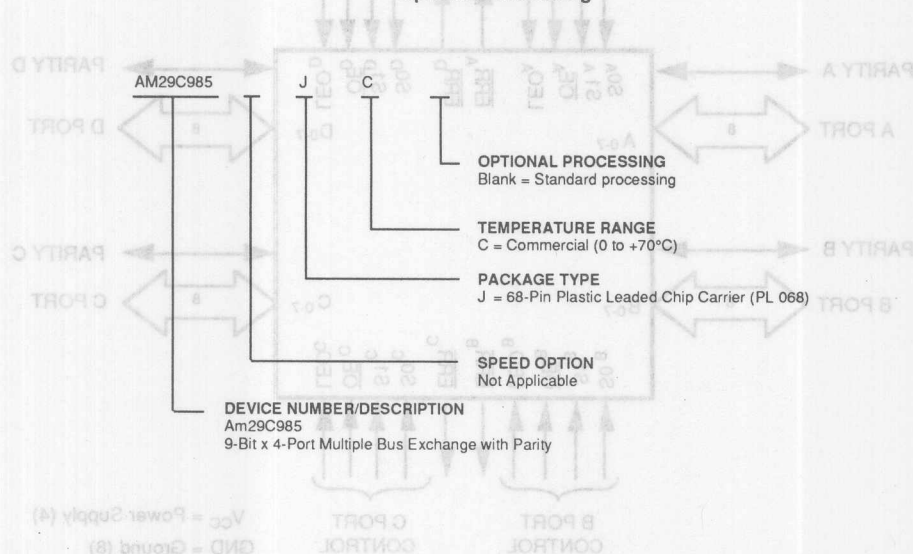
11996-005B

ORDERING INFORMATION

Standard Products

AMD standard products are available in several packages and operating ranges. The order number (Valid Combination) is formed by a combination of these elements:

Device Number
Speed Option (if applicable)
Package Type
Temperature Range
Optional Processing



Valid Combinations	
AM29C985	JC

Valid Combinations

Valid Combinations list configurations planned to be supported in volume for this device. Consult the local AMD sales office to confirm availability of specific valid combinations, to check on newly released combinations, and to obtain additional data on AMD's standard military grade products.

PIN DESCRIPTION

A_i, B_i, C_i, and D_i (i = 0 through 7) Data Bus I/O Ports (Input/Output)

These four groups of eight I/O pins are defined as the A, B, C, and D ports respectively. Each port serves as a source (Input) or as a destination (Output).

PARITY A, PARITY B, PARITY C and PARITY D Parity Flag (Input/Output, Three-state)

As an input, parity and port are combined and checked for odd parity. As an output, parity is an active output indicating odd parity for port.

Si_i, Si_b, Si_c, and Si_d (i = 0, 1) Source Port Select (Inputs)

Each pair of inputs determines the source of data for the corresponding I/O port when used as a destination port.

Am29C985 OPERATIONAL DESCRIPTION

Parity and bus checking are provided on the Am29C985. Parity checking and generation are both performed at the output. In order to preserve parity coverage through the part, the data driven onto the bus, including the generated parity, is compared to the data passing through the multiplexer, including the old parity. This has two effects: The comparison of the parity bits acts as a parity check. Also bus errors will be detected if the bus data does not agree with the data being driven.

Minimization of Ground Bounce through Output Edge-Rate Control

The Am29C985 incorporates AMD's proprietary edge controlled outputs in order to minimize simultaneous switching noise (ground bounce). By controlling the output transient currents, ground bounce and output ringing have been greatly reduced. A modified AMD output provides a stable, usable voltage level in less time than a controlled output.

Additionally, speed degradation due to increasing number of outputs switching is reduced. Together, these benefits of edge-rate control result in significant increase in system performance despite a minor increase in specified device propagation delay.

Power-Up/Down Disable

The Am29C985 contains a unique power up/down circuit to provide glitch free outputs during power-supply sequencing. This power-up circuit ensures that at low V_{CC} values (typically 0–2.0 V), the outputs are disabled and in 3-state. At V_{CC} values above this threshold, the outputs will remain disabled and not glitch to an active state if the appropriate output-

ERR_A, ERR_B, ERR_C, and ERR_D ERROR (Output, open drain)

Each output pin is used to flag Parity/Bus errors. Error is indicated by a LOW output.

LEO_A, LEO_B, LEO_C, and LEO_D Output Latch Enable (Inputs; Active HIGH)

Each LEO input controls a 9-bit wide latch on the output side of the corresponding I/O port. The latches are transparent when LEO is HIGH and are latched when LEO is LOW.

OE_A, OE_B, OE_C, and OE_D Output Enable (Inputs; Active LOW)

Each OE input controls the bus drivers of the corresponding I/O port. When OE is LOW, data at the output of the Output latches is passed to the bus. When OE is HIGH, the bus outputs are in high-impedance state.

enable inputs are conditioned for 3-state functionality. At V_{CC} values above the disable circuitry threshold, if the output-enable inputs are conditioned active (outputs enabled), the outputs will respond to a steady state input value. Additionally, the outputs will exhibit high impedance characteristics under power conditioning.

Input/Output Structures

Typical CMOS devices on the market today have maximum DC I/O voltage ratings that prevent some card edge applications, due to the uncertainty of the I/O voltage with respect to V_{CC} . This uncertainty occurs when extracting or replacing a card into a powered-on connector or when a powered-off device is sitting on an active bus. Under these conditions, the maximum rating of -0.5 V to $V_{CC} + 0.5$ V may be violated. This rating is derived from the presence of a parasitic diode from the input or output to V_{CC} . To prevent forward biasing the diode with an active signal, the 0.5 V limit above V_{CC} was adopted.

AMD has addressed this situation with unique input and output structures. These structures on the Am29C985 use an n-channel pull-up transistor. This results in a stacked n-channel output buffer and a proprietary ESD input cell.

These circuit modifications result in a maximum DC I/O voltage rating of -0.5 V to 7.0 V. The maximum rating is no longer a function of the V_{CC} voltage, thus allowing 3-state functionality under power off condition.

In addition, another benefit gained is that the n-channel pull-up reduces the output HIGH-level voltage for a lightly loaded output to 4.0 V, at $V_{CC} = 5.0$ Volts. This reduces switching noise and cross-talk associated with typical CMOS full rail-to-rail travel.

FUNCTIONAL DESCRIPTION

The Am29C985 Multiple Bus Exchange consists of four 9-bit I/O ports. Each port has a 9-bit output latch to capture outgoing data. There are four control pins associated with each port: two Select inputs for source port selection, one Output Latch Enable input (active HIGH) to control Output latches, and an active LOW Output Enable line to control the bus driver at the I/O port.

Port Selection and Control

Each port is independently controlled by these four control inputs. If the output drivers of a port are disabled (high-impedance state), that port is an input and can be used as a source port. At the same time, the data at one of the four internal buses can be transferred to the Output latch under the control of the appropriate Select inputs. If the output drivers are enabled, the port serves as a destination port, transporting the data at the output of its Output latch to the external bus connected to the I/O port. Independent control of the Output latch permits stored operation at any port.

Parity and Bus checking

In the Am29C985, parity checking and recognition are both performed at the output. To preserve parity coverage through the part, the data driven onto the bus, including the regenerated

parity, is compared to the data passing through the switch, including the old parity. This has two effects: the comparison of parity bits acts as a parity check. Also bus errors will be detected if the bus data does not agree with data being driven to the output buffer.

Error Outputs

ERR pins are active LOW, open drain outputs. This allows easy combination of multiple bytes. When passing non-parity data through the part the output will have correct odd parity, but an error may be indicated due to the uncertainty of the 9th bit. Under this condition it is up to the user to ignore the error.

Multiple Bus Communication

Four internal buses serve as pathways for port-to-port connection. By proper choice of source select codes for the ports, the Am29C985 can be configured in different modes for multiple bus communication. In one mode of operation, two ports can be selected as source ports and the other two as destination ports; thus, two independent bidirectional communication channels are established. In another mode, one port can be selected as the source, and one or more of the other ports can serve as destination ports. Any port not intended as a destination port can be disabled (high-impedance state) by its Output Enable control.

A. Port Source Selection

S _{1n}	S _{0n}	Source
L	L	A Bus
L	H	B Bus
H	L	C Bus
H	H	D Bus

B. Output Latch Operation

LEO _n	Mode
H	Transparent
L	Latched

C. I/O Port Controls

LEO _n	OE _n	I/O	Source of Data
L	L	Out	Contents of Output Latch
H	L	Out	Selected Source Port
X	H	In	

Key: n = A, B, C, or D
L = LOW
H = HIGH
X = Don't Care

ABSOLUTE MAXIMUM RATINGS

OPERATING RANGES

Supply Voltage (V_{CC})	-0.5 to 7.0 V	Commercial (C) Devices	0 to +70°C
DC Input Diode Current (I_{IK}) ($V_{IN} < 0$ V)	-20 mA	Temperature (T_A)	+4.5 to +5.5 V
($V_{IN} > V_{CC}$ if applicable)	+20 mA	Supply Voltage (V_{CC})	
DC Input Voltage (V_{IN})	-0.5 to 7.0 V		
DC Output Diode Current (I_{OK}) ($V_{OUT} < 0$ V)	-50 mA	<i>Operating ranges define those limits between which the functionality of the device is guaranteed.</i>	
($V_{OUT} > V_{CC}$ if applicable)	+50 mA		
DC Output Current per Output Pin:			
I_{SINK}	+70 mA		
I_{SOURCE}	-30 mA		
DC Output Voltage (V_{OUT})	-0.5 to 7.0 V		
Total DC Ground Current (I_{GND})	1750 mA		
Total DC V_{CC} Current (I_{CC})	575 mA		
Storage Temperature	-65 to +150°C		

Stresses above those listed under ABSOLUTE MAXIMUM RATINGS may cause permanent device failure. Functionality at or above these limits is not implied. Exposure to absolute maximum ratings for extended periods may affect device reliability.

DC CHARACTERISTICS over operating range unless otherwise specified.

Parameter Symbol	Parameter Description	Test Conditions		Min.	Max.	Unit
V_{OH}	Output HIGH Voltage	$V_{CC} = 4.5$ V $V_{IN} = V_{IL}$ or V_{IH}	$I_{OH} = -15$ mA	2.4		V
V_{OL}	Output LOW Voltage	$V_{CC} = 4.5$ V $V_{IN} = V_{IL}$ or V_{IH}	$I_{OL} = 48$ mA		0.5	V
V_{IH}	Input HIGH Voltage	(Note 1)		2.0		V
V_{IL}	Input LOW Voltage	(Note 1)			0.8	V
V_{IC}	Input Clamp Voltage	$V_{CC} = 4.5$ V, $I_{IN} = -18$ mA			-1.2	V
I_{IL}	Input LOW Current (Select Inputs)	$V_{CC} = 5.5$ V, $V_{IN} = 0$ V			-10	μA
I_{IH}	Input HIGH Current (Select Inputs)	$V_{CC} = 5.5$ V, $V_{IN} = 5.5$ V			10	μA
I_{OZL}	Off-State Leakage Current (I/O Ports)	$V_{CC} = 5.5$ V, $V_{OUT} = 0$ V			-20	μA
I_{OZH}	Off-State Leakage Current (I/O Ports)	$V_{CC} = 5.5$ V, $V_{OUT} = 5.5$ V			20	μA
I_{SC}	Output Short-Circuit Current	$V_{CC} = 5.5$ V, $V_{OUT} = 0$ V (Note 2)		-60		mA
I_{CCO}	Quiescent Power Supply Current (Note 4)	$V_{CC} = 5.5$ V, $V_{IN} = 5.5$ V or GND, Outputs Open		1.5		mA
I_{CCT}	Power Supply Current TTL Input HIGH (Note 4)	$V_{CC} = 5.5$ V, $V_{IN} = 2.4$ V Other Inputs at V_{CC} or GND		3.0		mA/ Input

DC CHARACTERISTICS (Cont'd.)

Parameter Symbol	Parameter Description	Test Conditions	Min.	Max.	Unit
I_{CCD}	Dynamic Power Supply Current (Note 4)	$V_{CC} = 5.5$ V, Outputs Open One Output Toggling (Note 3)		500	μ A MHz/Bit

SWITCHING CHARACTERISTICS over operating range unless other specified.

No.	Parameter Symbol	Parameter Description	Test Conditions	Min.	Max.	Unit
1	t_{PLH}	Propagation Delay Port to Port	$C_L = 50$ pF $R_1 = 500$ Ohms $R_2 = 500$ Ohms	1.5	12	ns
2	t_{PHL}	LEO = HIGH		1.5	12	ns
3	t_{PLH}	Propagation Delay Select Input to Port		1.5	12	ns
4	t_{PHL}	LEO = HIGH		1.5	12	ns
5	t_{PLH}	Propagation Delay Port to Parity		1.5	14	ns
6	t_{PHL}	LEO = HIGH		1.5	14	ns
7	t_{PLH}	Propagation Delay		1.5	10	ns
8	t_{PHL}	LEO to Port		1.5	10	ns
9	t_{PZH}	Output Enable Time		1	10	ns
10	t_{PZI}	\overline{OE} to Port		1	10	ns
11	t_{PHZ}	Output Disable Time		0	9	ns
12	t_{PLZ}	\overline{OE} to Port		0	9	ns
13	t_{PD}	Propagation Delay Port to \overline{ERR} Valid (Note 5)		2.0	14	ns
14	t_{PLH}	Propagation Delay		2.0	15	ns
15	t_{PHL}	Select to Parity		2.0	15	ns
16	t_{PLH}	Propagation Delay		2.0	14	ns
17	t_{PHL}	LEO to Parity		2.0	14	ns
18	t_s	Port to LEO Setup		4.5		ns
19	t_h	Port to LEO Hold		0		ns
20	t_s	Select to LEO Setup		6.0		ns
21	t_h	Select to LEO Hold		0		ns
22	t_{PWH}	LEO Pulse Width HIGH		3		ns

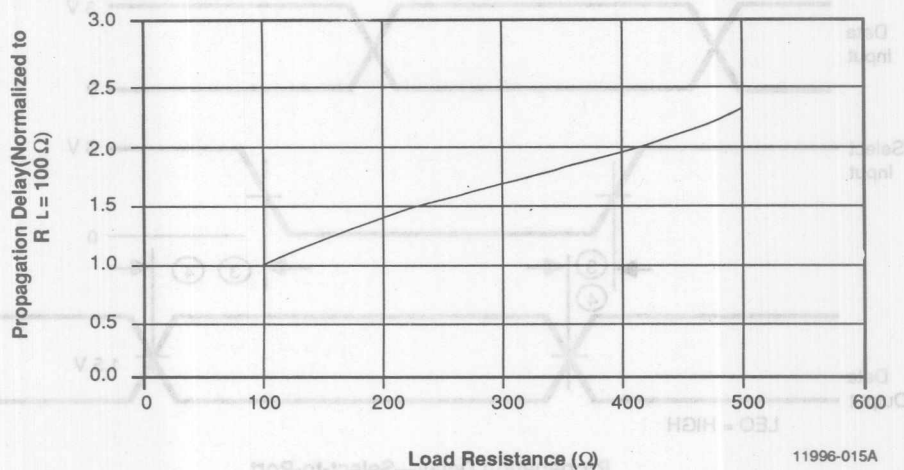
Notes:

- Input thresholds are tested in combination with other DC parameters or by correlation.
- Not more than one output shorted at a time. Duration of short-circuit test not to exceed 100 ms.
- Measured at a frequency of < 10 MHz with 50% duty cycle. Unused inputs are at V_{CC} or GND.
- Calculation of total device I_{CC} : $I_{CC} = I_{CC0} + I_{CCT}D_HN_T + I_{CCP}(f_{CP}/2 + f_{f_i})$
 Where: D_H = Duty cycle for each TTL input HIGH
 N_T = Number of inputs at D_H
 f_{CP} = Clock frequency for clocked devices (Zero for non-clocked devices)
 f_{f_i} = Input frequency of the i^{th} input
 N_i = Number of inputs at f_i
- The propagation delay time is proportional to the output pull-up resistor to V_{CC} . In this case the measurement is done with the pull-up resistor = 100 Ω . Please refer to graph on the following page for more details



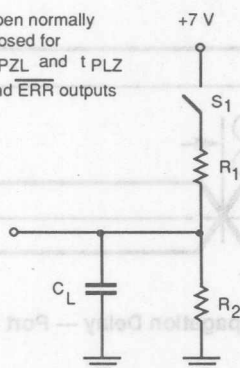
Am29C985 Port Out to $\overline{\text{ERR}}$ Valid

$V_{CC} = 4.50 \text{ V}$, $T_a = 70^\circ\text{C}$



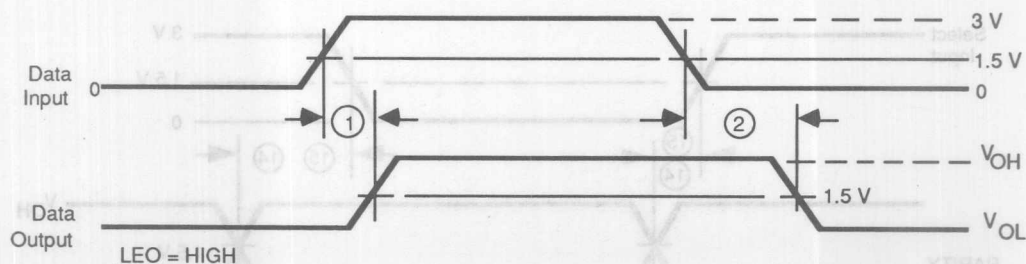
SWITCHING TEST CIRCUIT

S_1 = Open normally
 S_1 = Closed for t_{PZL} and t_{PLZ} and $\overline{\text{ERR}}$ outputs



11996-006A

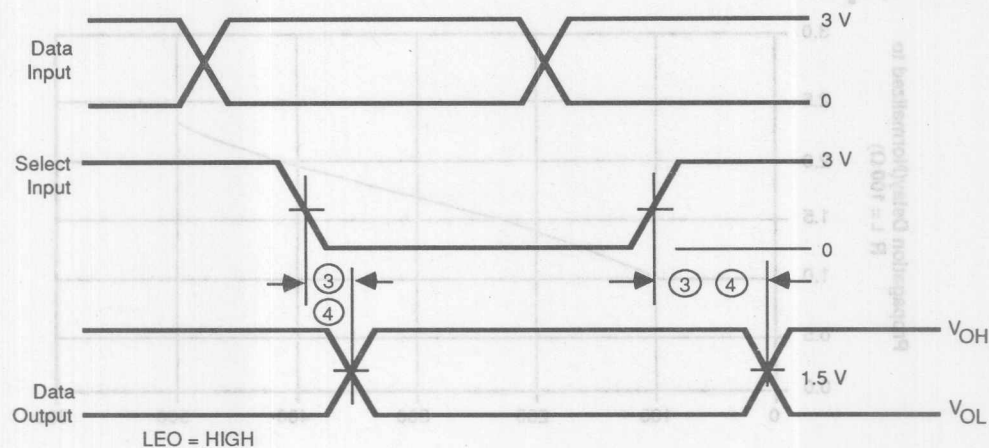
SWITCHING TEST WAVEFORMS



Propagation Delay—Port-to-Port

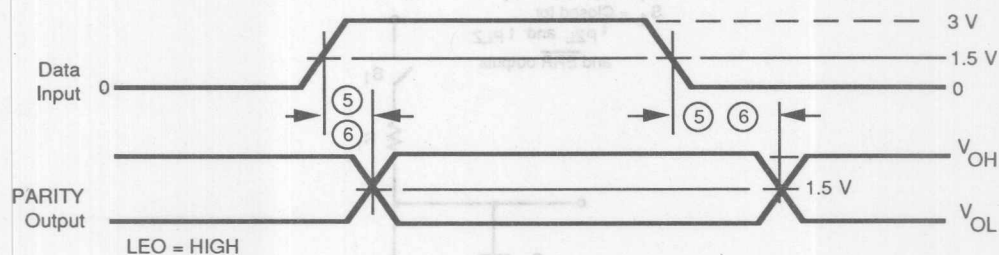
11996-007A

SWITCHING TEST WAVEFORMS (Cont'd.)



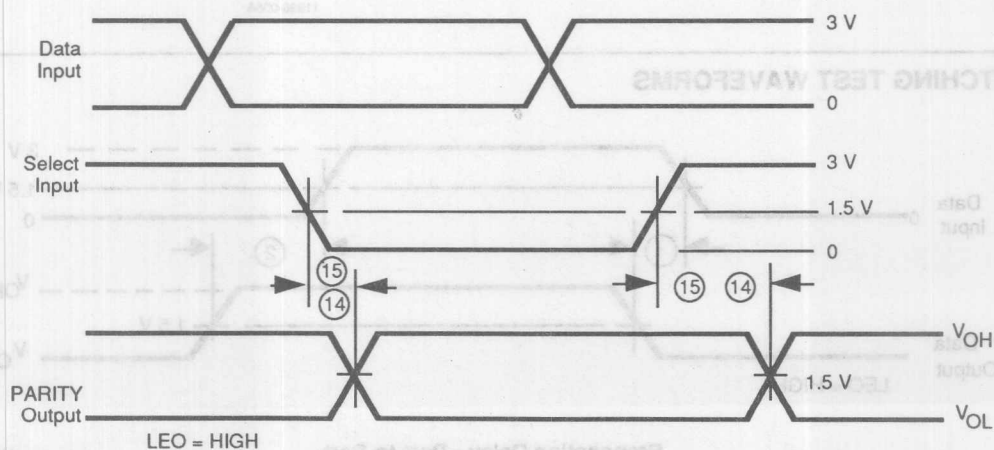
Propagation Delay—Select-to-Port

11996-008A



Propagation Delay — Port to Parity

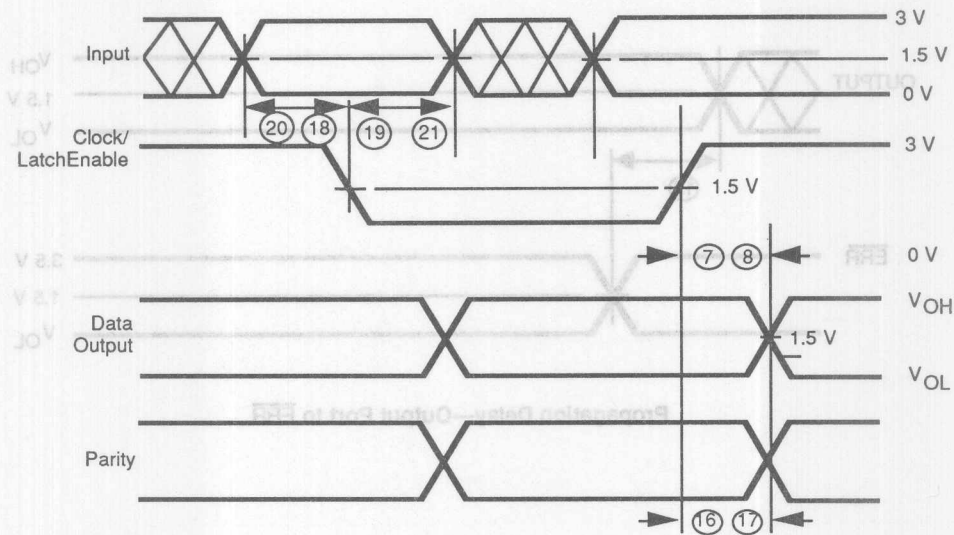
11996-009A



Propagation Delay — Select to Parity

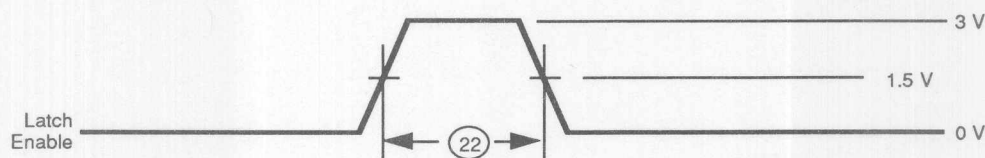
11996-010A

SWITCHING TEST WAVEFORMS (Cont'd.)



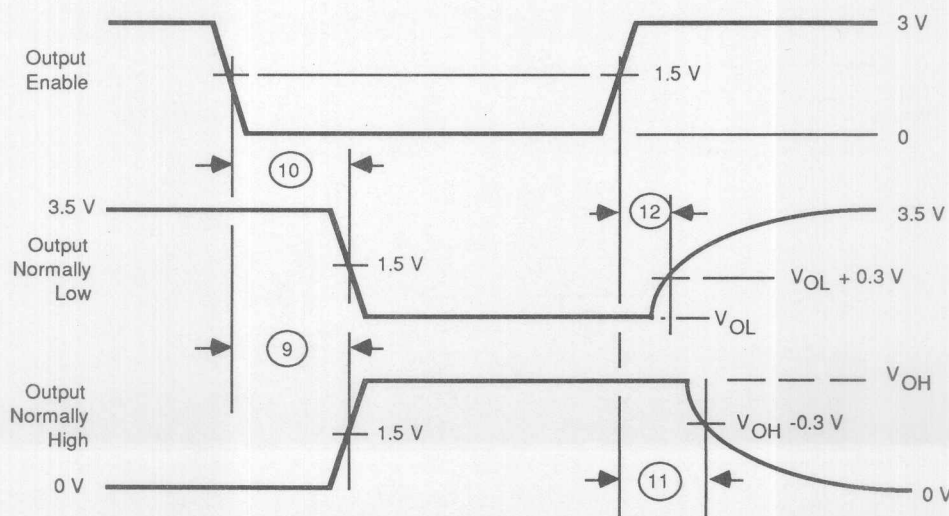
11996-011A

Output Latch Propagation Delay, Setup and Hold Times



11996-012A

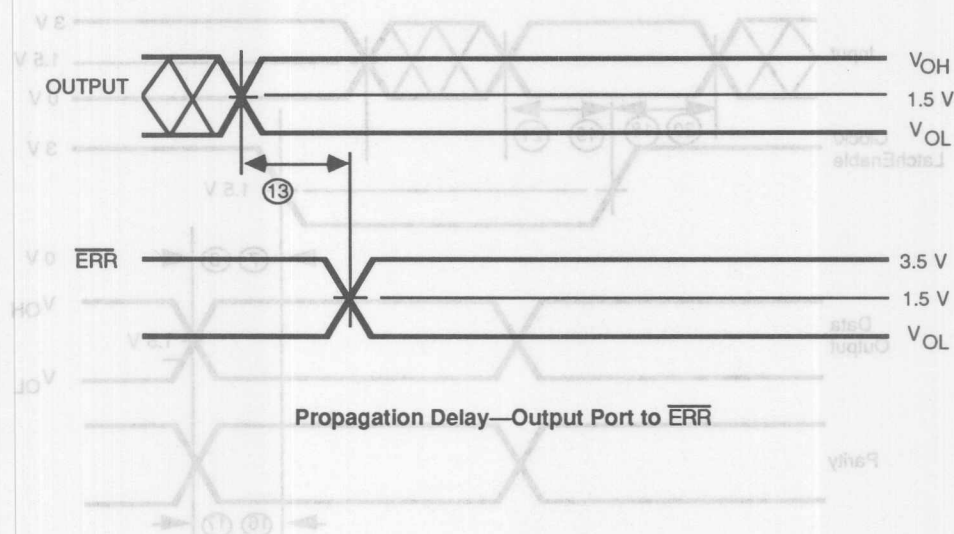
Minimum Latch Enable



11996-013A

Enable and Disable Times

SWITCHING TEST WAVEFORMS (Cont'd.)



11996-014A

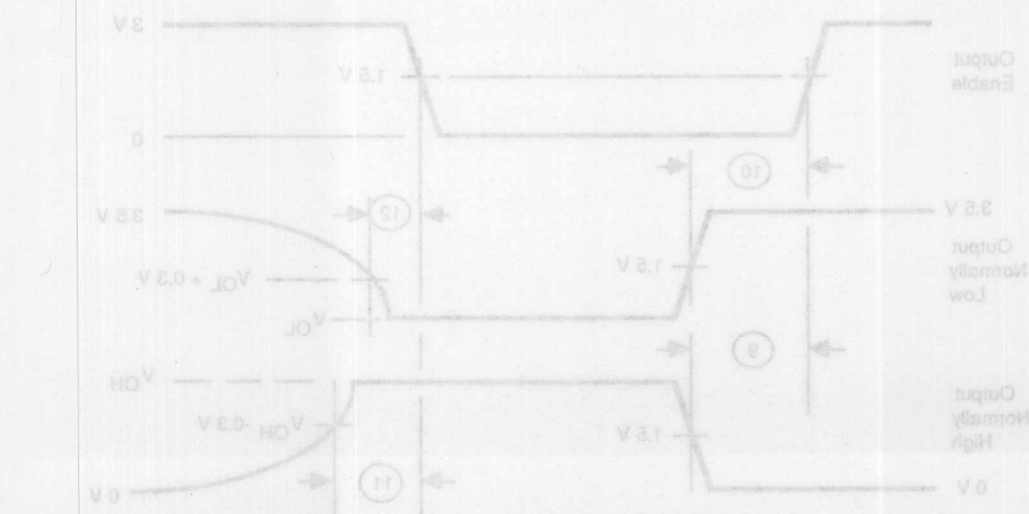
11996-014A

Output Latch Propagation Delay, Setup and Hold Times



11996-012X

Minimum Latch Enable



11996-014A

Enable and Disable Times

CHAPTER 7

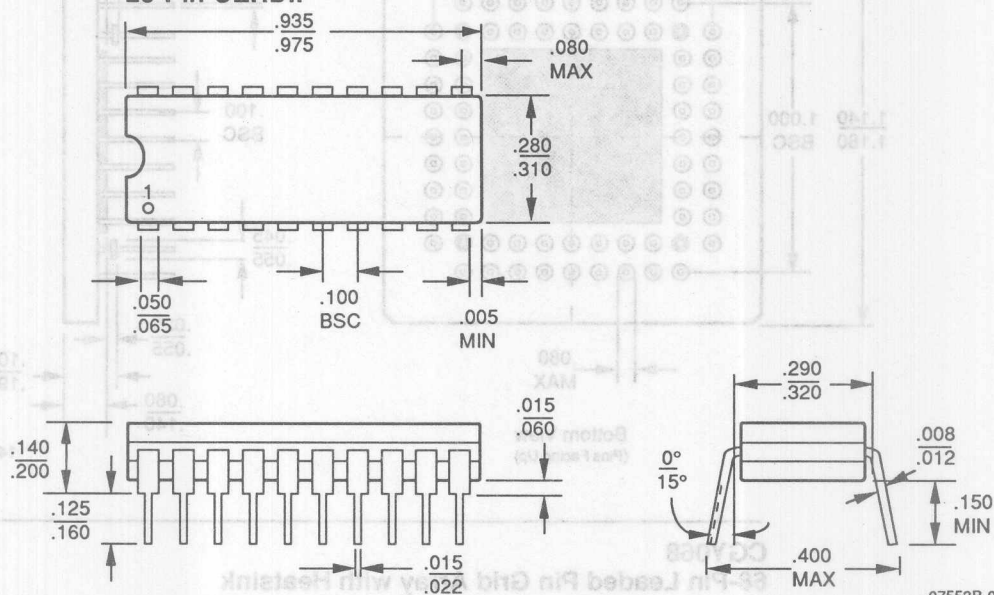
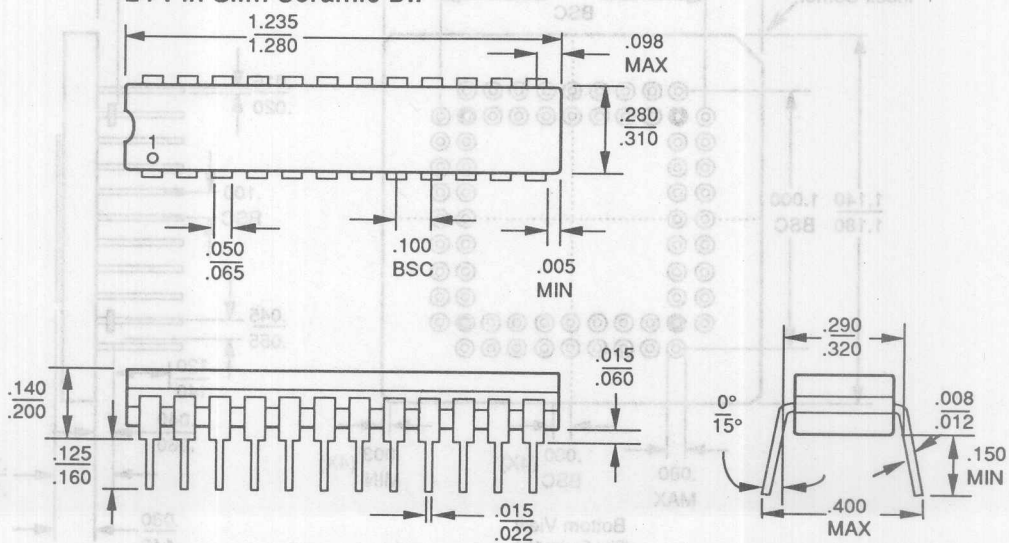
Physical Dimensions*



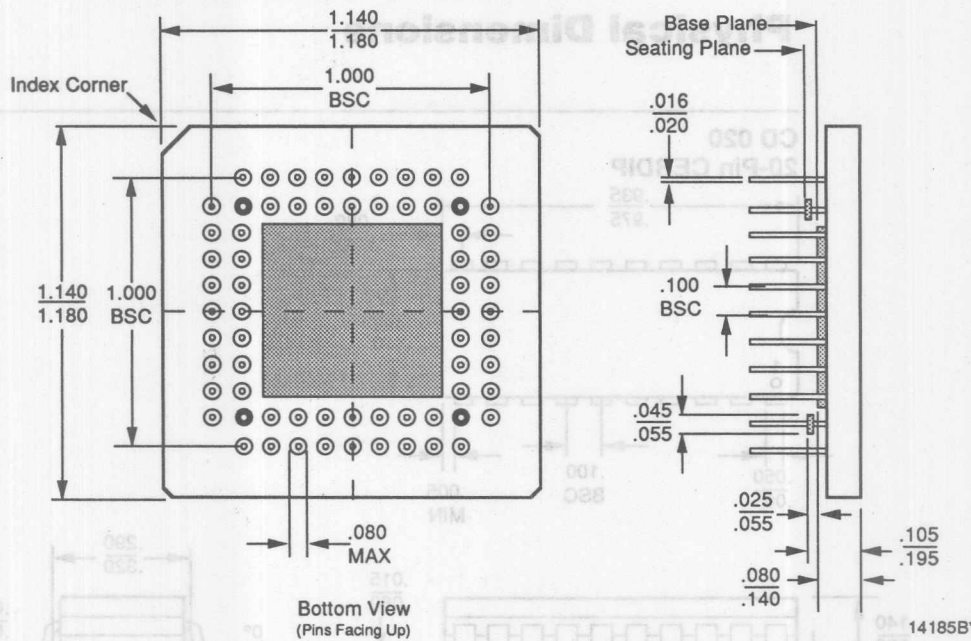
CD 020	20-Pin CERDIP
CD3024	24-Pin Slim Ceramic DIP
CGX 068	68-Pin Ceramic Pin Grid Array
CGY 068	68-Pin Leaded Pin Grid Array without Heatsink
PD 020	20-Pin Plastic DIP
PD 048	48-Pin Plastic DIP
PD3024	24-Pin Plastic DIP
PL 020	20-Pin Plastic Leaded Chip Carrier
PL 068	68-Pin Plastic Leaded Chip Carrier
PQJ80	80-Pin Plastic Quad Flat Pack Package
SD 048	48-Pin Sidebrazed Ceramic DIP
SO 020	20-Pin Plastic Small Outline Package
SO 024	24-Pin Plastic Small Outline Package
SO 028	28-Pin Plastic Small Outline Package

* For reference only. All dimensions are measured inches unless otherwise specified. BSC is an ANSI standard for Basic Space Centering.

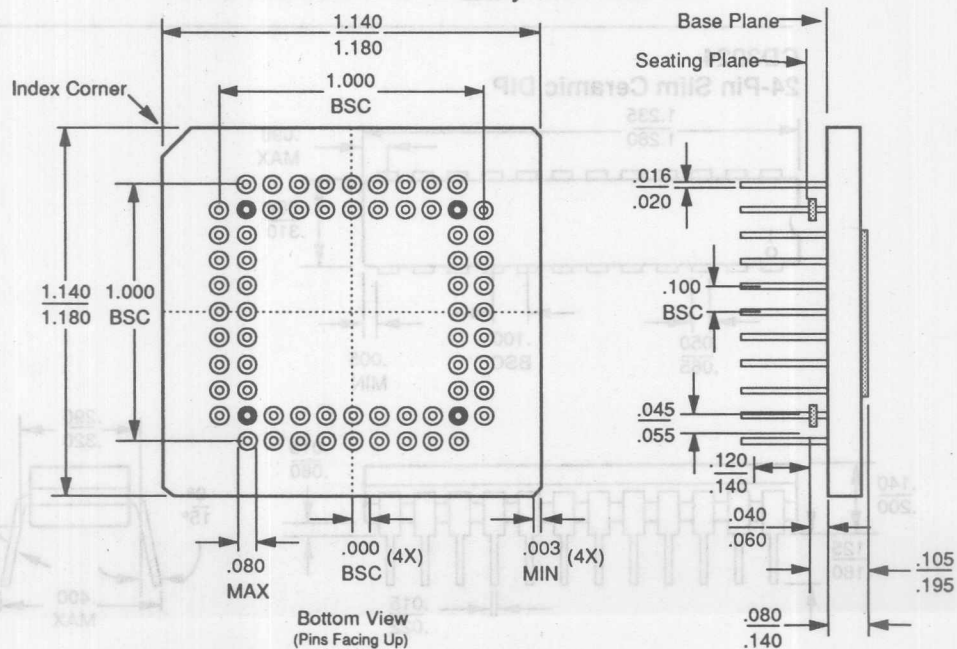
Physical Dimensions

CD 020
20-Pin CerdipCD3024
24-Pin Slim Ceramic DIP

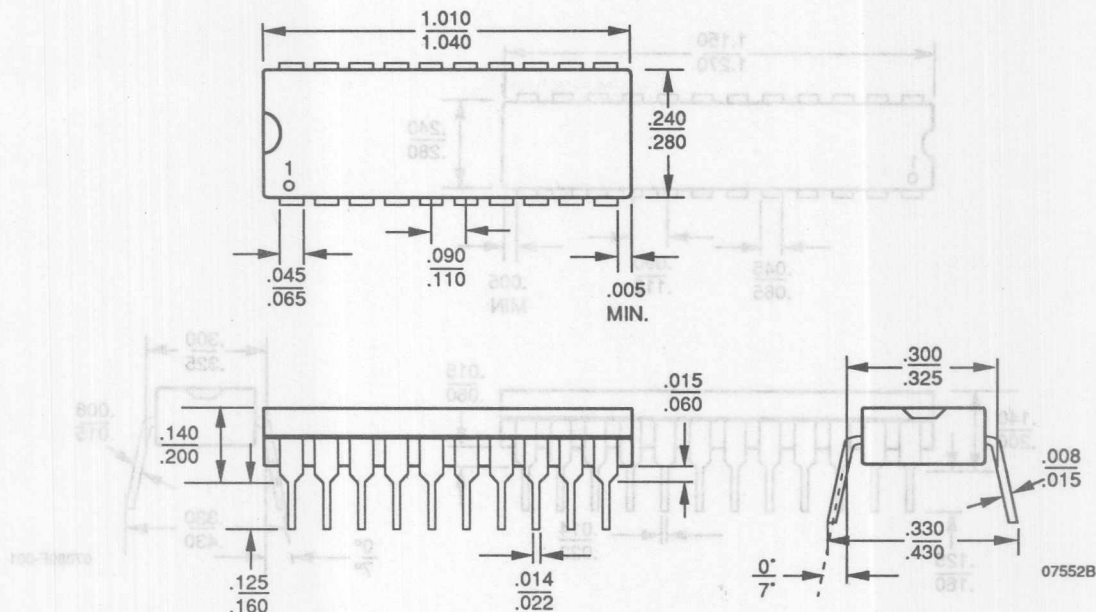
CGX068 68-Pin Ceramic Pin Grid Array



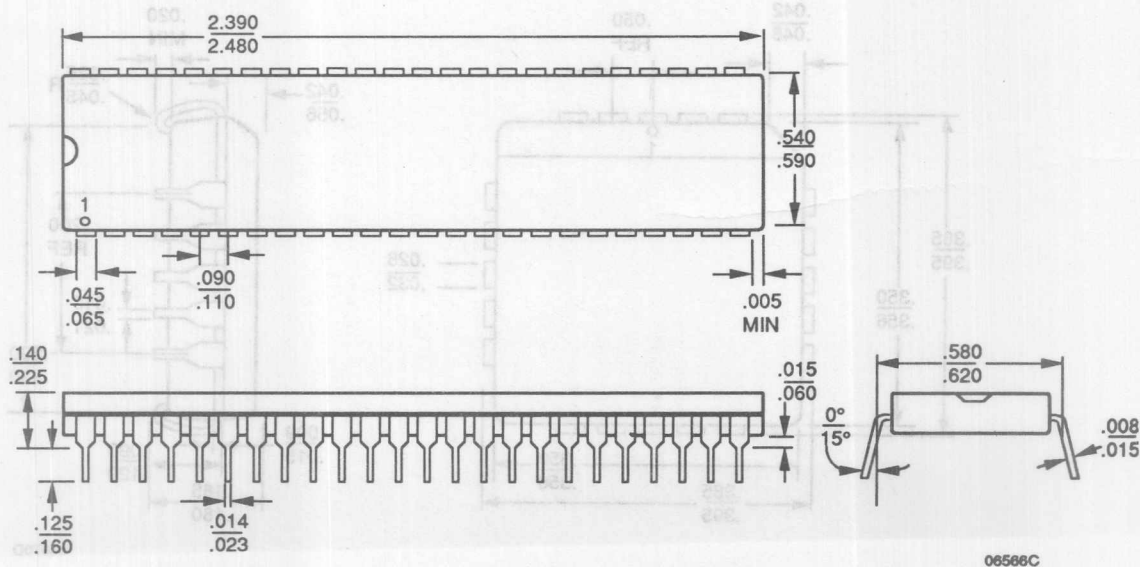
CGY068 68-Pin Leaded Pin Grid Array with Heatsink



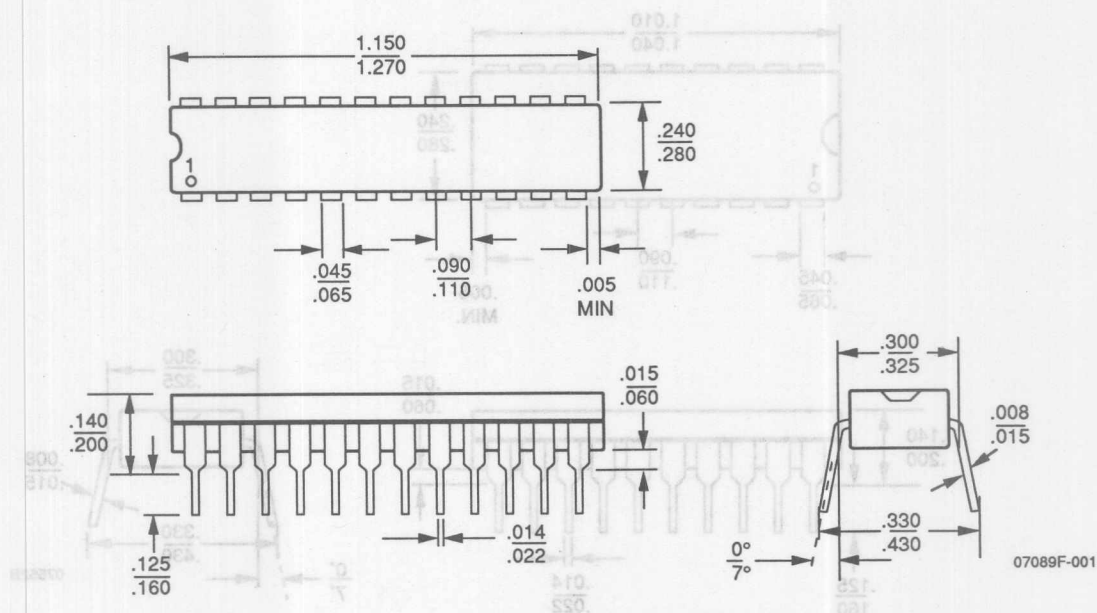
PD 020
20-Pin Plastic DIP



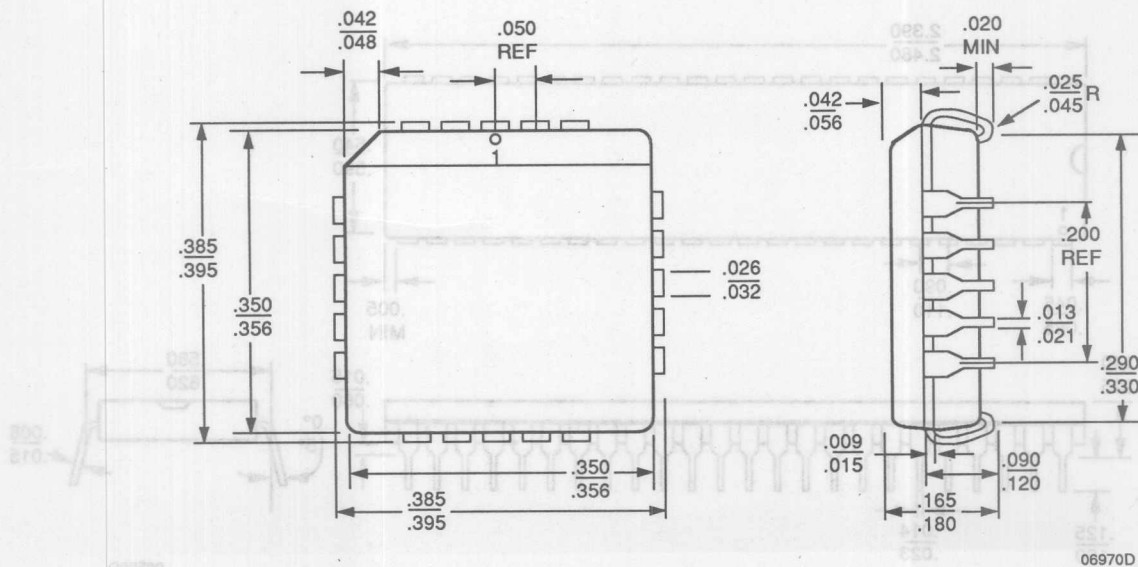
PD 048
48-Pin Plastic DIP



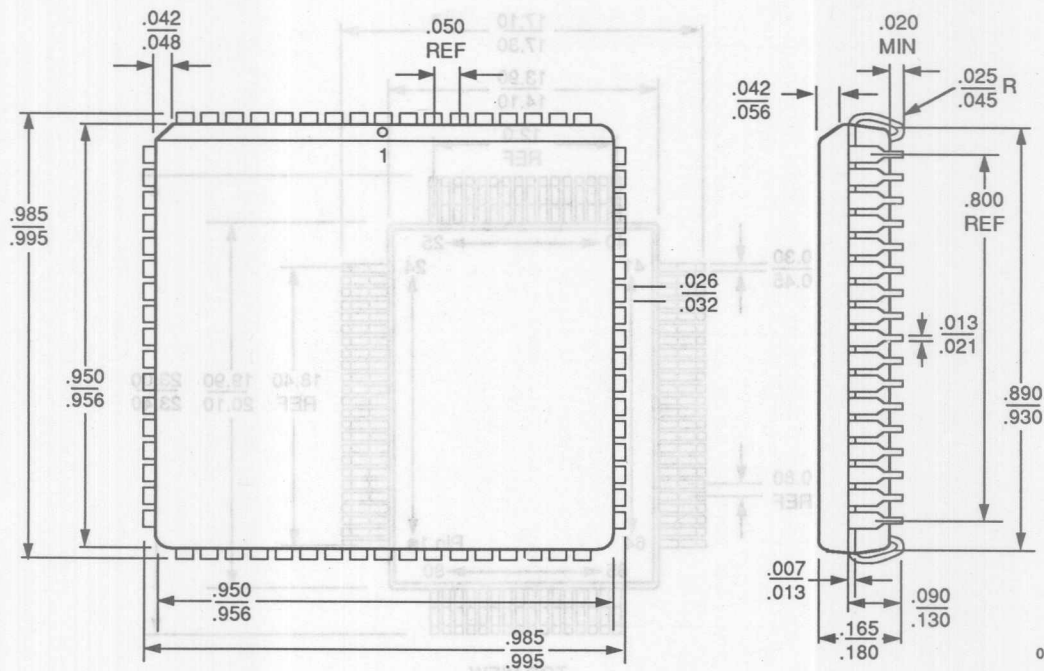
PD3024
24-Pin Plastic DIP



PL 020
20-Pin Plastic Leaded Chip Carrier

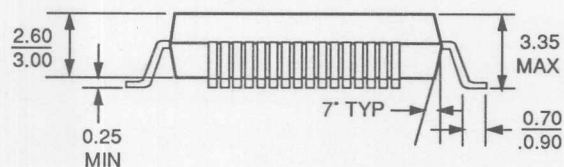
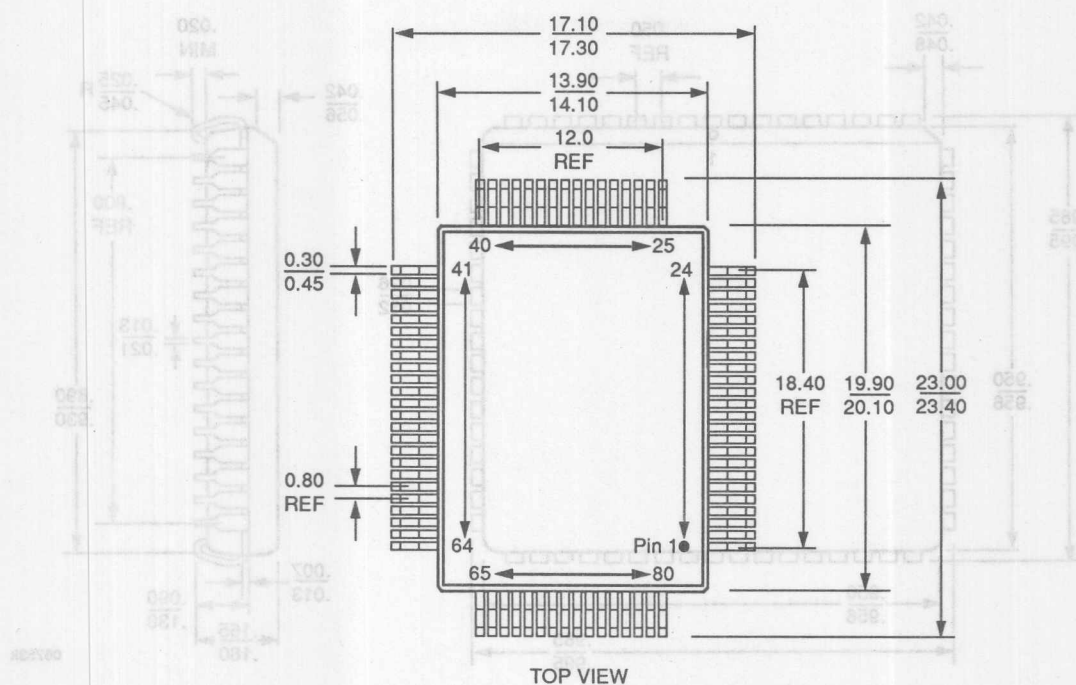


PL 068
68-Pin Plastic Leaded Chip Carrier



06753k

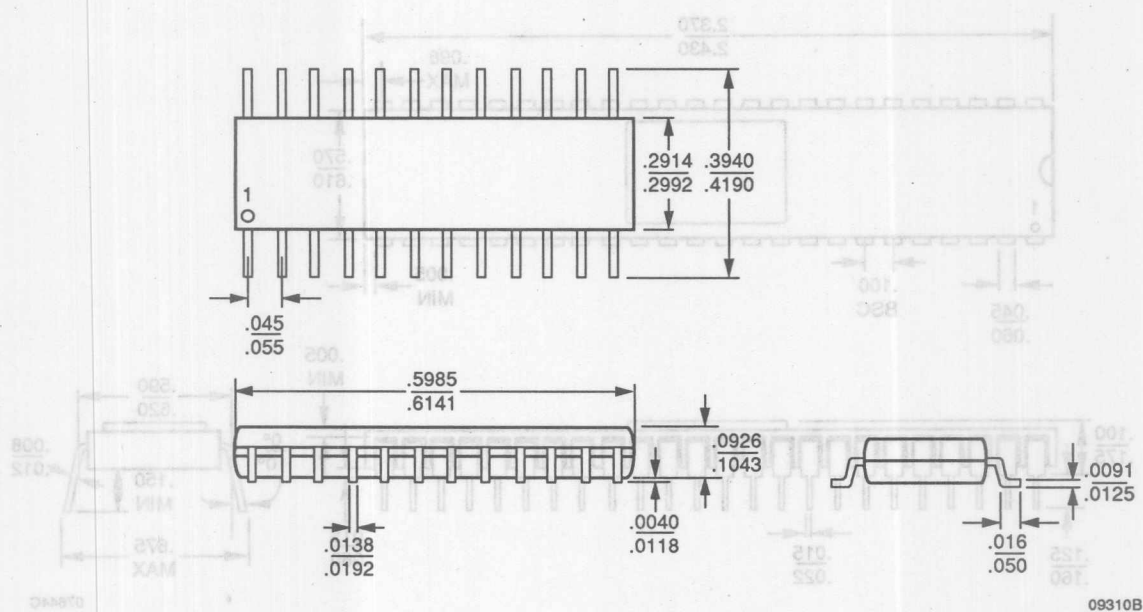
PQJ 80
80-Pin Plastic Quad Flat Pack Package



14635C

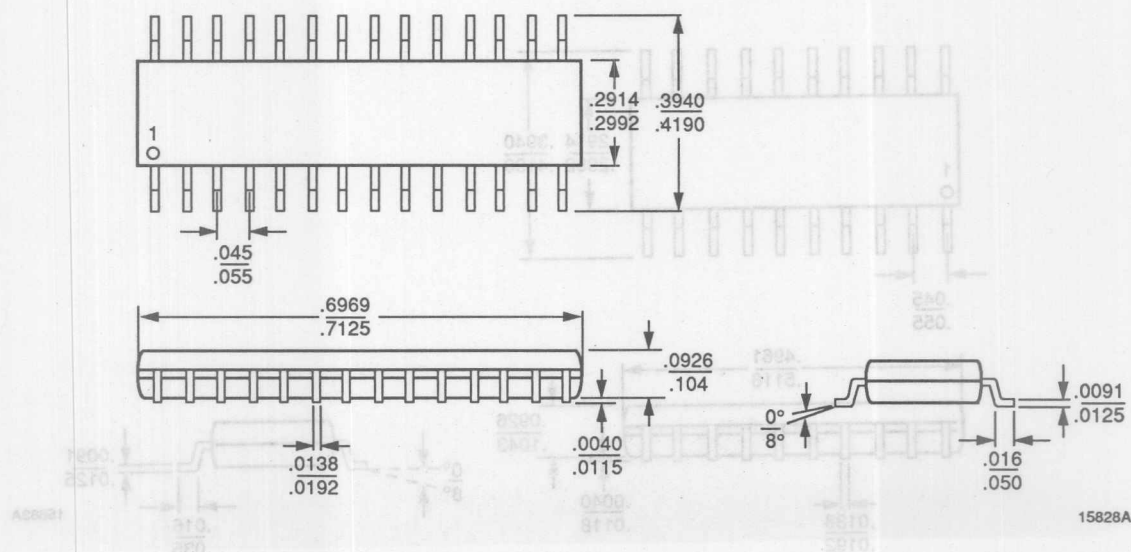
SO 024

24-Pin Plastic Small Outline Package



SO 028

28-Pin Plastic Small Outline Package





APPENDIX A

Behavioral Simulation Models From Logic Automation, Inc.

Behavioral Simulation Models From Logic Automation, Inc.	A-3
--	-----



APPENDIX A Behavioral Simulation Models From Logic Automation, Inc.

Behavioral Simulation Models From
Logic Automation, Inc.

A-1



Behavioral Simulation Models From Logic Automation, Inc.

- Fast and Accurate
- Extensive Usage and Timing Checks
- Over 5000 Devices Supported
- Compatible with Leading Simulators
- SmartModel Windows™, a System Emulation Capability for Viewing and Changing Register Contents

SmartModels™ are behavioral language-simulation models with built-in expert assistance, used for board- and system-level simulation. Models of thousands of devices are available ranging from complex microprocessors to memories, PLDs, and TTL logic. Simulation provides several benefits to the user including faster design time, lower prototype costs, and higher quality product.

SmartModels increase designer productivity with extensive messages including usage checks and timing checks. The SmartModel usage checks look for undefined interrupts, uninitialized registers, illegal conditions—any misuse of the component that is likely to slow or stall the design process. These are reported as thoroughly as possible, pin-pointing the error by documenting the design sheet, part instance, pin name, and the time of occurrence, so the error can be eliminated immediately.

SmartModel timing checks look for violations of timing specifications like set-up, hold, and recovery. Messages cite the required specification as well as the violation, along with the pin location and simulation time. The result is that the designer need not interrupt the system verification to search component data books for specifications. The necessary data is right there, built into the simulation models.

High Performance Bus Interface devices supported include (for product information see Bus Interface Product Data Book, PID 11128B):

- Am29821, Am29823, Am29825, Am29827, Am29828, Am29841
- Am29818A, Am29827A, Am29833A, Am29853A, Am29861A
- Am29C818A, Am29C821A, Am29C823A, Am29C827A, Am29C828A
- Am29C833A, Am29841A, Am29C843A, Am29C853A, Am29C861A, Am29C863A

Multiple Bus Exchange devices supported include (for product information see MBE Handbook, PID 10315B):

- Am29C982
- Am29C983, Am29C983A
- Am29C985

Dynamic Memory Management devices supported include:

- Am29C668, Am29C668-1, Am29C676
- Am29C660, Am29C660A, Am29C660B, Am29C660C, Am29C660D, Am29C660E
- Am29C60, Am29C60-1, Am29C60A
- Am29C676

FIFO Interface devices supported included (for product information see Memory Products Data Book, PID 07606C):

- Am7202A, Am7203A, Am7204A, Am7205A

Host Systems supported now include:

- Mentor Graphics
- Valid Logic
- Gateway Design Automation
- HHB Systems
- Vantage
- Hewlett Packard
- AT&T (proprietary)
- GENRAD
- Cadence Design Systems
- Racal-Redac

Host Systems under development include:

- DAZIX
- Teradyne
- Viewlogic
- Silicon Compiler Systems

The following factory contacts at Logic Automation may be contacted for price and availability information:

Corporate Headquarters:

Logic Automation Incorporated
19500 NW Gibbs Drive
P.O. Box 310
Beaverton, OR 97075
Phone: (503) 690-6900
FAX: (503) 690-6906
Electronic Bulletin Board: (503) 690-6907

European Sales Office:

Logic Automation (Europe), Ltd.
Stratfield House
265 High Street
Crowthorne
Berkshire RG117AH
United Kingdom
Phone: (0) 344 778822
FAX: (0) 344 775703

Japan:

Logic Automation
KSP R&D C-4F
100-1 Sakado
Takatsu-Ku, Kawasaki-shi
Kangawa Pref, 213 Japan
Phone: (044) 812-7420
FAX: (044) 812-7421

France:

Logic Automation
1, Boulevard de l'oise
95050 Cergy Cedex France
Phone: (1) 34-22-30-32
FAX: (1) 30-38-45-70

Asia Headquarters:

Logic Automation Incorporated
4010 Moorpark Avenue, Suite 105
San Jose, CA 95117
Phone: (408) 247-1442
FAX: (408) 249-3767



APPENDIX B

Electronic Design Automation Tools From OrCAD Systems Corporation

Electronic Design Automation Tools From OrCAD Systems Corporation	B-3
--	-----



APPENDIX B Electronic Design Automation Tools From OrCAD Systems Corporation

Electronic Design Automation Tools From
OrCAD Systems Corporation

B-3



Electronic Design Automation Tools From OrCAD Systems Corporation

With today's complex designs, the efficiency of Electronic Design Automation products is more than important, it's vital. Your company doesn't have the time or resources to redraw schematics, revise part lists and netlists using bad tools. For this reason, AMD has developed OrCAD™ models of our most popular High Performance Bus Interface, Multiple Bus Exchange and Dynamic Memory Management products.

OrCAD is the world's largest volume producer of Electronic Design Automation tools. OrCAD/STD III is quick and easy to learn. Intuitive pop-up menus and quick keyboard commands let you start designing immediately. Work in OrCAD/STD III flows logically in a progression of steps mirroring your own intuitive approach to design.

OrCAD/STD III includes everything you need to handle the most complex design challenges on your PC. In the package are powerful ways to automate:

- Bill of materials/parts list generation
- Electrical Rules Checking
- Forward and backward annotation
- Cross reference
- Netlist generation

High Performance Bus Interface devices supported include (for product information see Bus Interface Product Data Book, PID 11128B):

- Am29821, Am29823, Am29825, Am29827, Am29828, Am29841
- Am29818A, Am29827A, Am29833A, Am29853A, Am29861A
- Am29C818A, Am29C821A, Am29C823A, Am29C827A, Am29C828A
- Am29C833A, Am29C841A, Am29C843A, Am29C853A, Am29C861A, Am29C863A

Multiple Bus Exchange devices supported include (for product information see MBE Handbook, PID10315B):

- Am29C982
- Am29C983, Am29C983A
- Am29C985

Dynamic Memory Management devices supported include:

- Am29C668, Am29C668-1, Am29C676
- Am29C660, Am29C660A, Am29C660B, Am29C660C, Am29C660D, Am29C660E
- Am29C60, Am29C60-1, Am29C60A
- Am29C676

FIFO Interface devices supported include (for product information see Memory Products Data Book, PID 07606C):

- Am7202A, Am7203A, Am7204A, Am7205A

OrCAD models of AMD products are available by contacting OrCAD or by logging onto AMD's bulletin board system. To log onto AMD's bulletin board system via modem, please dial one of the following numbers:

(408) 744-4659 or (408) 744-4346

Corporate Headquarters:

OrCAD Systems Corporation

Hillsboro, Oregon 97124

Phone: (503) 690-9881

FAX: (503) 690-9891

Electronic Bulletin Board: (503) 690-9791